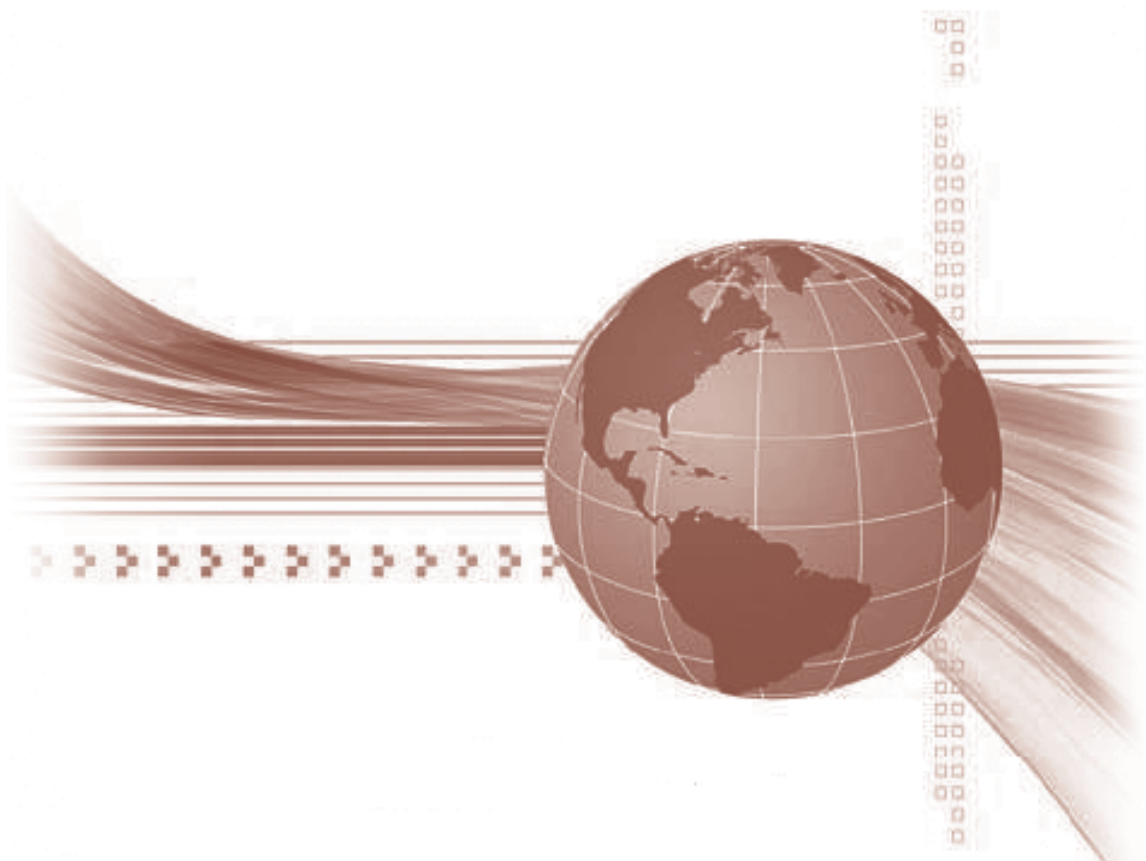
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

Special Issue 2/2014

STUDIA

UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

Special Issue 2/2014

June

EDITORIAL BOARD

YEAR **Volume 59 (LIX) 2014**
MONTH **JUNE**
ISSUE **SPECIAL ISSUE 2**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

SPECIAL ISSUE 2:
12TH INTERNATIONAL CONFERENCE ON FORMAL CONCEPT ANALYSIS – ICFCA 2014

EDITORIAL OFFICE: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Special Issue 2, 2014
ICFCA 2014: 12th International Conference on Formal Concept Analysis, Cluj-Napoca, June 10-13, 2014

ICFCA 2014: THE 12TH INTERNATIONAL CONFERENCE ON FORMAL CONCEPT ANALYSIS

CYNTHIA VERA GLODEANU, MEHDI KAYTOUE, AND CHRISTIAN SĂCĂREA

Formal Concept Analysis (FCA) is a multi-disciplinary field built on the solid foundation of lattice and order theory. Besides this, FCA is strongly rooted in philosophical aspects of the mathematical formalization of concept and concept hierarchy. Since its emergence in the 1980s the field has developed into a constantly growing research area in its own right, with a thriving theoretical community further applying and developing this powerful framework of qualitative analysis of data. One of the initial goals of FCA was to promote better communication between lattice theorists and potential users of lattice theory. The increasing number of applications in diverse areas such as data visualization, information retrieval, data mining and knowledge discovery demonstrates how that goal is being met.

In order to offer researchers the opportunity to meet and discuss developments and applications of FCA annually, the International Conference on Formal Concept Analysis (ICFCA) was established and held for the first time in Darmstadt, Germany in 2003. Since then, the ICFCA has been held in different countries from Europe, America, Africa and in Australia.

The 12^{th} ICFCA took place from the 10^{th} to the 13^{th} of June, 2014 at the Babeş-Bolyai University, Cluj-Napoca, Romania. There were 39 submissions by authors from 14 different countries. Each paper was reviewed by 3 members of the Program Committee (exceptionally four). Sixteen high-quality papers were chosen for publication in the conference proceedings volume, amounting to an acceptance rate of 41%. Six other works in progress were considered valuable for presentation during the conference, five of them being included in this special issue of the journal *Studia Universitatis Babeş-Bolyai, Series Informatica*.

We were also delighted that three prestigious researchers accepted to give an invited talk:

- *Learning Spaces, and How to Build them* by Prof. Jean-Paul Doignon, Université Libre de Bruxelles, Belgium;
- *On the Succinctness of Closure Operator Representations* by Prof. Sebastian Rudolph, Technische Universität Dresden, Germany;

- *MDL for Pattern Mining: A Brief Introduction to Krimp* by Prof. Arno Siebes, Universiteit Utrecht, The Netherlands.

Our deepest gratitude goes to all the authors of submitted papers. Choosing ICFCA 2014 as a forum to publish their research was key to the success of the conference. Besides the submitted papers, the high quality of the published volumes would not have been possible without the strong commitment of the authors, the Program Committee and Editorial Board members, and the external reviewers. Working with the efficient and capable team of local organizers was a constant pleasure. We are deeply indebted to all of them for making this conference a successful forum on FCA.

Last, but not least, we are most grateful to the editorial board of *Studia Universitatis Babeş-Bolyai, Series Informatica* for accepting to publish the *Contributions to ICFCA 2014* as a special issue of this journal, as well to the organizations that sponsored this event: the Bitdefender company, the iQuest company, the Babeş-Bolyai University, and the City of Cluj-Napoca, Romania. Finally, we would like to emphasize the great help of EasyChair for making the technical duties easier.

Technische Universität Dresden, Zellescher Weg 12-14, 01062 Dresden, Germany
E-mail address: Cynthia-Vera.Glodeanu@tu-dresden.de

INSA de Lyon, 20 Avenue Albert Einstein, 69621 Villeurbanne, France
E-mail address: mehdi.kaytoue@insa-lyon.fr

Babeş-Bolyai University, Kogălniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: csacarea@cs.ubbcluj.ro

Organization

Executive Committee

Conference Chair
Christian Sacarea, Babeş-Bolyai University, Cluj-Napoca, Romania

Conference Organizing Committee
Brigitte Breckner, Babeş-Bolyai University, Cluj-Napoca, Romania
Sanda Dragoş, Babeş-Bolyai University, Cluj-Napoca, Romania
Diana Haliţă, Babeş-Bolyai University, Cluj-Napoca, Romania
Diana Troancă, Babeş-Bolyai University, Cluj-Napoca, Romania
Viorica Varga, Babeş-Bolyai University, Cluj-Napoca, Romania

Program and Conference Proceedings

Program Chairs
Cynthia Vera Glodeanu, Technische Universität Dresden, Germany
Mehdi Kaytoue, Université de Lyon, France

Editorial Board
Peggy Cellier, IRISA, INSA Rennes, France
Felix Distel, Technische Universität Dresden, Germany
Florent Domenach, University of Nicosia, Cyprus
Peter Eklund, University of Wollongong, Australia
Sebastien Ferré, Université de Rennes 1, France
Bernhard Ganter, Technische Universität Dresden, Germany
Robert Godin, Université du Québec à Montréal,Canada
Robert Jäschke, Leibniz Universität Hannover, Germany
Sergei O. Kuznetsov, Higher School of Economics, Russia
Leonard Kwuid, Bern University of Applied Sciences, Switzerland
Rokia Missaoui, Université du Québec en Outaouais, Canada
Sergei Obiedkov, Higher School of Economics, Russia
Uta Priss, Ostfalia University of Applied Sciences, Germany
Sebastian Rudolph, Technische Universität Dresden, Germany
Stefan E. Schmidt, Technische Universität Dresden, Germany
Gerd Stumme, University of Kassel, Germany
Petko Valtchev, Universite du Québec à Montréal, Canada
Karl Erich Wolff, University of Applied Sciences, Germany

Honorary Member
Rudolf Wille, Technische Universität Darmstadt, Germany

Sponsoring Institutions

The Babeş-Bolyai University Cluj-Napoca, Romania
The City of Cluj-Napoca, Romania
The Bitdefender Company, Bucharest, Romania
iQuest GmbH & Co KG, Frankfurt, Germany

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Special Issue 2, 2014
ICFCA 2014: 12th International Conference on Formal Concept Analysis, Cluj-Napoca, June 10-13, 2014

CLASSIFYING SOFTWARE BUG REPORTS USING METHODS FROM FORMAL CONCEPT ANALYSIS

DANIEL BORCHMANN, RAFAEL PEÑALOZA, AND WENQIAN WANG

Abstract. We provide experience in applying methods from formal concept analysis to the problem of classifying software bug reports characterized by distinguished features. More specifically, we investigate the situation where we are given a set of already processed bug reports together with the components of the program that contained the corresponding error. The task is the following: given a new bug report with specific features, provide a list of components of the program based on the bug reports already processed that are likely to contain the error. To this end, we investigate several approaches that employ the idea of *implications* between features and program components. We describe these approaches in detail, and apply them to real-world data for evaluation. The best of our approaches is capable of identifying in just a fraction of a second the component causing a bug with an accuracy of over 70 percent.

1. Motivation

Maintaining large software systems is a non-trivial task, and processing bug reports efficiently is a crucial part of this process. Modern software systems can easily contain thousands of lines of code, distributed over several modules and subsystems. When the system reaches such a size, and no single programmer can oversee its overall complexity, finding components of the program which are likely to contain the error causing a given bug report becomes much more demanding. This is a known challenge in software development. For example, a recent study showed that in average it takes 19 days for the Eclipse project and 38 days for the Mozilla project to find a first component assignment for a bug report [6], without guarantee that this first assignment is correct. Finding the responsible component is a main bottleneck in the debugging process, and it may even require more time than fixing the error itself. In

Received by the editors: March 26, 2014.
2010 *Mathematics Subject Classification.* 68N30, 03G10.
1998 *CR Categories and Descriptors.* K.6.3 Software Management-Software maintenance.
Key words and phrases. FCA, Classification, Software Maintenance, Implications.
D. Borchmann supported by DFG Graduiertenkolleg 1763 (QuantLA). R. Peñaloza partially supported by DFG within the Cluster of Excellence 'cfAED'.

such cases, speeding up the process of identifying the responsible components would increase maintainability, and thus the quality, of the software.

The purpose of this work is to share some experimental experience we have obtained while trying to solve this problem. The approaches we follow in this work are all based on ideas from formal concept analysis. More precisely, we employed the idea that the information contained in a bug report (its so-called "features") somehow determine in an *implicational manner* the component of the program containing the error. Therefore, we devised several methods based on the notion of *implications* in formal contexts to find such components, and tried to evaluate them experimentally on some real-world data obtained from bug reports in a large software company.

Obviously, one could argue here that the assumption that features of bug report determine the responsible components precisely is somehow simplified: it is not unlikely—and it is in fact not very hard to come up with an example for this—that two identical bug reports are caused by two completely unrelated errors in the software system. Clearly, this defeats our main assumption of an implicational dependency between features of bug report and responsible components. On the other hand, one could argue that the cause for such situation is that the bug reports are *under-specified*, and that the implicational dependencies between features of bug report and responsible components would still hold if we would include more features, which add information that can separate the two reports. This could be achieved by requesting more information from the user reporting the bug. However, even in the case where we do not request additional information, we can still use our assumption to find a *set* of likely components that caused the bug report, thus reducing the number of components which need to investigated.

The main practical problem we have to face when following the indicated approaches is to *find* the implicational dependencies between features of bug report and their responsible components. To cope with this difficulty, our approaches more or less follow a common pattern: all bug reports already processed so far are brought together in a formal context $\mathbb{K}_{\mathsf{reports}}$. This formal context is then examined for implicational dependencies between features and components. Then, if a new bug report, given as a set of features, is received, the implications extracted from the initial context are applied to this set of features, and the components contained in the resulting closure are considered *candidate causes* for the new bug report. Additionally, some of our approaches introduce a meaningful way of *rating* the candidate components according to their likelihood; that is, the higher the rank of a candidate component, the more likely it is that the new bug was caused in that component.

While this idea is relatively simple to describe and understand, it faces several practical issues. As already discussed, if the set of features does not

determine responsible components uniquely, the standard approaches of extracting valid implications from the context $\mathbb{K}_{\mathsf{reports}}$ are not applicable. Hence, we must devise new methods to achieve this goal, relaxing the restrictions that implications must satisfy. Furthermore, already processed bug reports do not always need to be correct; for example, it may happen that the actual cause of some historical report was never fixed, but rather that the circumstances of the bug were altered in such a way that it was not observed any more. In such cases, the component stored as cause for this error in the historical records is itself not correct. Assuming that such cases are possible but unlikely, we have to adapt our approaches to include methods that can handle those exceptional errors correctly. Finally, the context $\mathbb{K}_{\mathsf{reports}}$ itself can be quite large, and existing approaches to extract implications from contexts may simply not work on such large contexts, due to memory and time restrictions. Devising ideas for scalable extraction algorithms is thus also necessary in this setting.

The problem of suggesting components that are likely responsible for bug reports is a classification problem in the sense of machine learning [11]. However, it is not the aim of this work, at this early stage of development, to compete with existing methods from machine learning. Our purpose is more to share experiences on how to approach this problem from the perspective of formal concept analysis, which we consider a natural, although often neglected, choice for this situation. A comparison with other existing classification approaches, or a combination with them, would be a logical next step in this direction of research. We leave that road open for possible future work.

This work is organized as follows. After giving a formal specification of our problem and some related work in Section 2, we introduce and discuss in Section 3 the approaches we investigate in this paper. Thereafter, we describe our experimental setup, show and discuss our results, and evaluate the individual approaches. This is done in Section 4. We close this paper with conclusions and outlook for further research in Section 5.

2. Problem Specification and Related Work

We first describe the problem we want to solve in a more precise way. For the rest of this paper, we assume familiarity with the basic notions of formal concept analysis. More details from this area can be found in [5].

Let $\mathbb{K}_{\mathsf{reports}} = (G, M, I)$ be a finite formal context, called the *context of reports*, and let $M = F \cup C$ be a partition of M, i. e. $F \cap C = \emptyset$ and F, C are non-empty. We call the elements of F *features*, and those of C *components*. Intuitively, we understand $\mathbb{K}_{\mathsf{reports}}$ as the formal context of all *previous issues* (*old issues*, or *bug reports*) that have been reported for our software system. For every such issue $g \in G$, the elements of $g' \cap F$ are the *features* of the issue

of g; i.e., the information observed and reported when the user encountered the error. Possible such features can be statements like "segmentation fault" or "screen turned blue". On the other hand, the elements of $g' \cap C$ are the *responsible components* of the issue g, i. e. the elements of the software that produces the issue g, and were located when the old issue g was solved. In other words, fixing these components resulted in the issue to disappear.

Given such a formal context $\mathbb{K}_{\mathsf{reports}}$ and the partition $M = F \cup C$, we want to find for a given new issue (that is, for a set of features $\mathsf{o} \subseteq F$) a set of components which are "likely" to be responsible for it. To achieve this goal, we want to make use of the historical knowledge from the already solved issues collected in $\mathbb{K}_{\mathsf{reports}}$. Thus, we want to be able to *learn* from the old issues as a means to identifying the components that are responsible for a new issue.

From this formalization of our problem, one may be reminded of a similar approach to model *learning from positive and negative examples* within FCA [8]. In this approach we assume a formal context $\mathbb{L} = (H, N, J)$, and a *target attribute* $\omega \notin M$ which objects in H may or may not have. Let $H_+ \subseteq H$ be the set of objects which are known to have the attribute ω, $H_- \subseteq H$ the set of objects that do not have the attribute ω and let $H_? = H \setminus (H_+ \cup H_-)$ be the set of objects for which it is not known whether they have the attribute ω or not. The three sets H_+, H_-, and $H_?$ are mutually disjoint. We call the elements of H_+ *positive examples* for ω, and likewise elements of H_- *negatives examples* for ω. The elements of $H_?$ are called *undetermined examples*.

The sets $H_+, H_-, H_?$ give rise to three subcontexts $\mathbb{L}_+, \mathbb{L}_-, \mathbb{L}_?$ of $\mathbb{L}$ defined as the restrictions of $\mathbb{L}$ to the corresponding sets of objects. The derivation operators of $\mathbb{L}_+, \mathbb{L}_-, \mathbb{L}_?$ are denoted by $(\cdot)^+, (\cdot)^-, (\cdot)^?$, respectively.

To decide for objects in $H_?$ whether they may have the target attribute ω or not, we extract *hypotheses* from $\mathbb{L}_+$ and $\mathbb{L}_-$. A *positive hypothesis* T for ω is an intent of $\mathbb{L}_+$ such that $T^+ \neq \emptyset$ and T is not contained in any object intent of $\mathbb{L}_-$, i. e. $T \not\subseteq g^-$ for all negative examples $g \in H_-$. *Negative hypotheses* are defined analogously. To decide for an undetermined example $g \in H_?$ whether it has the target attribute ω or not, we consider its object intent $g^?$ in the context $\mathbb{L}_?$. If this set contains positive hypotheses but no negative ones, then g is *classified positively*, and correspondingly, if $g^?$ contains negative hypotheses but no positive ones, g is *classified negatively*. If $g^?$ does not contain any hypotheses at all, then g is *unclassified*, and if $g^?$ contains both positive and negative hypotheses, then the classification of g is *contradictory*.

This method could also be applied to our problem of classifying software issues. In this case, we would consider every component we have as a target attribute, and try to apply the above method to obtain a classification. However, this idea becomes impractical as the number of components increases: for

TABLE 1. The context $\mathbb{K}_{\mathsf{exa}}$ used as a running example

object	a	b	c	X	Y
1	×	×		×	
2	×	×			×
3			×	×	
4	×	×	×	×	
5		×			×
6	×		×	×	

each component we would need to construct the contexts $\mathbb{L}_+, \mathbb{L}_-, \mathbb{L}_?$ and classify using the method sketched above, which is actually known to be hard [7]. This theoretical hardness may or may not be an issue in practical applications.

Furthermore, it may happen that bug reports having the exact same features, actually describe different errors in the software, and thus may have different responsible components. In those cases, we would still like to obtain a meaningful set of potentially responsible components (if possible, with an associated rating). However, the approach for learning from examples [8] would always result in an undetermined or contradictory classification.

Nevertheless, we can draw some inspiration from this approach for our own problem, and we do so in the following section, where we describe some methods for proposing responsible components for new issues.

3. Method Descriptions

We have tried several approaches for detecting the responsible components for a given issue. Each of these approaches is motivated by different ideas, which we describe in detail next. Their common property is that they all make use of a historical collection of old issues stored in the context $\mathbb{K}_{\mathsf{reports}}$ of reports to predict the component of a new issue. After having described these methods, in the next section we provide the results of an experimental evaluation on real-world issues from a software company.

For the following descriptions we assume that the attribute set M of $\mathbb{K}_{\mathsf{reports}}$ is partitioned into features and components as described before, i. e. $M = F \cup C$. Furthermore, we assume that we are given a *new issue* $\mathsf{o} \subseteq F$ which we want to classify. For this, each of the following methods *proposes* a set $\mathsf{candidates}(\mathsf{o}) \subseteq C$ of the components that are likely to be responsible for the issue o. Furthermore, all but the first method additionally yield a score $\mathsf{score}(x) \in [0, 1]$ for each component $x \in \mathsf{candidates}(\mathsf{o})$. The higher this score, the more likely the method considers x to be responsible for o.

To help understanding the ideas behind all these methods, we will apply them over the simple context $\mathbb{K}_{\mathsf{exa}}$ shown in Table 1. In this context, the

features are a, b, and c, while the components are X and Y. As the new issue to be classified we consider the set of features $\mathsf{o}_{\mathsf{exa}} = \{b, c\}$.

The `new-incident` method. A very simple idea for classifying a new issue would be to search in the historical records $\mathbb{K}_{\mathsf{reports}}$ for a previous occurrence of the same issue. The component that was responsible for the old issue can then be suggested as being responsible also for the new issue. This idea has two obvious problems. On one hand, the historical record is not necessarily complete, and hence there might exist no matching report; in this case, no responsible component would be suggested. On the other hand, since historical records may contain errors, components might change over time, and the set of features might not fully describe all possible problems, there might exist more than one matching issue in $\mathbb{K}_{\mathsf{reports}}$, which may lead to several components being proposed. To alleviate these issues, we slightly generalize this simple classification idea, yielding an approach which we call `new-incident`, which works as follows. Recall that the new issue is described by its set of features $\mathsf{o} \subseteq F$. For every object g in the context $\mathbb{K}_{\mathsf{reports}}$, if $g' \cap F \subseteq \mathsf{o}$, then $g' \cap C$ is suggested as a responsible component, i. e.

$$\mathsf{candidates}(\mathsf{o}) = \{x \in g' \cap C \mid g' \cap F \subseteq \mathsf{o}\}.$$

Note that there is no scoring among the candidates of o, i. e. all proposed components are equally preferred.

In our example context $\mathbb{K}_{\mathsf{exa}}$, for the new issue $\mathsf{o}_{\mathsf{exa}}$ we get that only the objects 3 and 5 are such that $g' \cap F \subseteq \mathsf{o}_{\mathsf{exa}}$: $3' \cap F = \{c\}$, and $5' \cap F = \{b\}$. The proposed components are then X and Y, and these are preferred equally.

This approach is in fact very similar to the one using hypotheses for classification, as we have described in Section 2. Namely, what we do here is to consider for all components $x \in C$ in $\mathbb{K}_{\mathsf{reports}}$ the sets of features belonging to issues in $\mathbb{K}_{\mathsf{reports}}$ with responsible component x. These sets actually correspond to hypotheses in the sense of Section 2. The only difference may be that for one such set of features T it may happen that T is actually contained in some set of features which belongs to a previous issue which had a responsible component different from x. Then, in the approach of Section 2 we would discard T as a hypothesis. However, as we have already argued previously, that is not a wanted behavior in our setting, as otherwise we would end up with a large number of contradictory classifications. Instead, we keep T as a hypothesis, and allow for a classification to more than one component. In this way, `new-incident` is similar to the classification of Section 2.

A drawback of the `new-incident` method is that the whole context needs to be processed whenever a new issue arises. As the historical records can be very large, this might be a very time-consuming task. Thus, we analyze

methods based on the pre-computation of *bases* of implications to assist in a more efficient classification of issues.

The `can+lux` method. Recall that the reports context may contain contradictory information or may be incomplete. It thus makes sense to try to use a base capable of producing implications that are violated by a small number of exceptions, like Luxenburger's base [9, 10, 14]. The definition of this base relies on the notions of *support* and *confidence* of an implication [1]. Intuitively, the support describes the proportion of objects that satisfy the implication, while the confidence measures the number of objects that do not violate it.

Definition 1 (support, confidence)**.** Let $\mathbb{K} = (G, M, I)$ be a formal context and $A \subseteq M$. The *support* of A is

$$\mathsf{supp}(A) := \frac{|A'|}{|G|}.$$

The *support* and *confidence* of an implication $A \to B$ are defined as

$$\mathsf{supp}(A \to B) := \mathsf{supp}(A \cup B), \qquad \mathsf{conf}(A \to B) := \frac{\mathsf{supp}(A \cup B)}{\mathsf{supp}(A)}.$$

Luxenburger's base includes only implications between intents having support and confidence larger than the given parameters $\mathsf{minsupp}$ and $\mathsf{minconf}$, respectively, which are input values from the interval [0,1] provided by the user. Moreover, the implications belonging to this base can only relate direct neighbors from the lattice of intents of the given formal context.

Definition 2 (Luxenburger's base)**.** For a finite formal context $\mathbb{K}$, the *Luxenburger base* of $\mathbb{K}$ w.r.t. $\mathsf{minsupp}, \mathsf{minconf} \in [0, 1]$ is the set of all implications $A \to B$ such that A and B are intents of $\mathbb{K}$, A is a direct lower neighbor of B in the lattice of intents of $\mathbb{K}$ ordered by $\subseteq$, and both (i) $\mathsf{conf}(A \to B) \geq \mathsf{minconf}$ and (ii) $\mathsf{supp}(A \to B) \geq \mathsf{minsupp}$ hold.

Notice that Luxenburger's base does not include implications that are valid in the formal context $\mathbb{K}$, because for two intents A, B of $\mathbb{K}$ to yield a valid implication $A \to B$ of $\mathbb{K}$, one must have $A = B$, and then A cannot be a direct lower neighbor of B anymore. To ensure that we do not miss implicational dependencies which are actually true in $\mathbb{K}_{\mathsf{reports}}$ we therefore have to take the valid implications separately, and we do so by extending Luxenburger's base with the *canonical base* of $\mathbb{K}_{\mathsf{reports}}$.

Given a new issue defined by a set of features o, the `can+lux` method computes the closure of o over the canonical and Luxenburger's bases of $\mathbb{K}_{\mathsf{reports}}$, and suggests all components appearing in this closure as candidates. Each candidate component $x \in C$ is associated with a score, defined as the maximum

of the confidences of all rules $A \to \{\, x \,\}$ such that $A \subseteq \mathsf{o}$, i. e.

$$\mathsf{score}(x) := \max\{\mathsf{conf}(A \to \{x\}) \mid A \subseteq \mathsf{o}\}.$$

Note that this involves an exhaustive search among all subsets of o, and can hence become very expensive. However, for the experimental setup that we discuss in the next section this is not an issue, as the size of o is usually small.

Let us consider our example context $\mathbb{K}_{\mathsf{exa}}$ again. Its canonical base is

$$\{\, \{Y\} \to \{b\}, \{b, X\} \to \{a\}, \{c\} \to \{X\}, \{a, b, Y, X\} \to \{c\} \,\},$$

and the Luxenburger's base of $\mathbb{K}_{\mathsf{exa}}$ with $\mathsf{minsupp} = 0.01$ and $\mathsf{minconf} = 0.01$ consists of the implications

$$\begin{array}{rrrr}
\emptyset \to \{X\}, & \emptyset \to \{b\}, & \{b\} \to \{Y\}, & \{X\} \to \{c\}, \\
\emptyset \to \{a\}, & \{X\} \to \{a\}, & \{a\} \to \{X\}, & \{b\} \to \{a\}, \\
\{a\} \to \{b\}, & \{a, X\} \to \{b\}, & \{a, b\} \to \{X\}, & \{b, Y\} \to \{a\}, \\
\{a, b\} \to \{Y\}, & \{c, X\} \to \{a\}, & \{a, X\} \to \{c\}, & \{a, b, X\} \to \{c\}, \\
\{a, c, X\} \to \{b\}. & & &
\end{array}$$

The closure of our observation $\mathsf{o}_{\mathsf{exa}} = \{b, c\}$ over these two bases includes both components X, Y, and hence both are proposed as responsible. Since the rule $\{c\} \to \{X\}$ is in the canonical base, X is proposed with score 1, while Y is proposed with score $\frac{2}{4}$, which is the confidence of the rule $\{b\} \to \{Y\}$.

The `can+lux` method provides a higher degree of liberty, as it is parameterized on the minimal support and minimal confidence that are used to compute Luxenburger's base. Moreover, the time required for computing the closure of the two bases and the scores of each proposed component is neglectable. Unfortunately, the same is not true for the computation of the bases. Indeed, as we will see in the following section, this computation was very costly in terms of time in our software issue scenario. Moreover, the performance of this classification was, surprisingly, rather disappointing.

Since the approach of considering Luxenburger's base turned out to be inappropriate, we studied different approaches for producing implications that are tolerant to a few exceptions. The main idea of the following three methods is to partition the context into smaller pieces, and compute only valid implications in these subcontexts. The intuition is that a small number of exceptions will violate such implications in only a few of all the subcontexts.

The `subcontext` method. For this method, we first create one subcontext $\mathbb{K}_x$ for every component $x \in C$ appearing in $\mathbb{K}_{\mathsf{reports}}$. The context $\mathbb{K}_x$ is defined as the restriction of $\mathbb{K}_{\mathsf{reports}}$ to the set of objects x' and thereafter removing all components and attributes which have an empty extent. In other words,

$$\mathbb{K}_x = (\bar{G} := x', \bar{F} := \{\, m \in F \mid m' \cap x' \neq \emptyset \,\}, I \cap \bar{G} \times \bar{F}).$$

TABLE 2. The subcontexts $\mathbb{K}_X$ (left) and $\mathbb{K}_Y$ (right) from `subcontext`

object	a	b	c
1	×	×	
3			×
4	×	×	×
6	×		×

object	a	b	c
2	×	×	
5		×	

Intuitively, the intents from the context $\mathbb{K}_x$ are sets of attributes that are always together whenever component x is responsible, and can hence be used as a premise for suggesting this component. To handle exceptions, we consider only implications whose premise have a support larger than a threshold, which is provided as a parameter. Formally, K is a *frequent intent* of a context $\mathbb{K}$ w.r.t. $\mathsf{minsupp}$ if it is an intent of $\mathbb{K}$ and $\mathsf{supp}(K) \geq \mathsf{minsupp}$. For every component x, and every frequent intent K of $\mathbb{K}_C$, we include the implication $K \to \{x\}$. Notice that every intent L that is a subset of a frequent intent K is also a frequent intent. Thus, it suffices to consider only the *minimal* frequent intents as premises for the implications. The proposed components are then

$$\mathsf{candidates}(\mathsf{o}) = \{x \mid K \text{ frequent non-empty intent of } \mathbb{K}_x, K \subseteq \mathsf{o}\}.$$

Note that this is the same as considering all components in the closure of o under all implications $K \to \{x\}$ with K a frequent, non-empty intent of $\mathbb{K}_x$.

Consider again our example context $\mathbb{K}_{\mathsf{exa}}$. The two subcontexts $\mathbb{K}_X$ and $\mathbb{K}_Y$ are shown in Table 2. If we set the minimal support to $\mathsf{minsupp} = 0.1$, then the minimal frequent intents of $\mathbb{K}_X$ are $\{a\}$ and $\{c\}$, and the only minimal frequent intent of $\mathbb{K}_Y$ is $\{b\}$. Thus, we obtain the rules $\{a\} \to X$, $\{b\} \to Y$, and $\{c\} \to X$. Given the new issue $\mathsf{o}_{\mathsf{exa}} = \{b, c\}$, both components X and Y are suggested as potentially responsible for the issue.

To provide a more fine-grained suggestion of the responsible components, we score these implications according to their relevance among the context of reports. More precisely, for each component we set

$$\mathsf{score}(x) := \max\{\mathsf{conf}(K \to \{x\}) \mid K \text{ frequent non-empty intent of } \mathbb{K}_x\}.$$

In our example, the scores are:

$$\mathsf{score}(X) = \max\{\mathsf{conf}(\{a\} \to X), \mathsf{conf}(\{c\} \to X)\} = 1,$$
$$\mathsf{score}(Y) = \max\{\mathsf{conf}(\{b\} \to Y)\} = \frac{2}{3}.$$

As a result, component X is suggested with a higher priority (1) than Y ($\frac{2}{3}$).

The `partition` and `partition-pp` methods. A different method for partitioning the context of all historical reports is to divide it in several subcontexts of equal size, regardless of the component they are associated with. Under the

TABLE 3. A partition $\mathbb{K}_1, \mathbb{K}_2$ of $\mathbb{K}_{\mathsf{exa}}$

object	a	b	c	X
1	×	×		×
3			×	×
6	×		×	×

object	a	b	c	X	Y
2	×	×			×
4	×	×	×	×	
5		×			×

assumption that exceptions occur rarely, we expect these exceptions to violate the implications in only a few of the generated subcontexts. As the first step in the `partition` and `partition-pp` methods, we randomly partition $\mathbb{K}_{\mathsf{reports}}$ into contexts of a specified size n. These subcontexts are then simplified by removing all attributes that appear in no object. For instance, the context $\mathbb{K}_{\mathsf{exa}}$ can be partitioned into two contexts of size 3 as shown in Table 3.

In the first context we have removed the attribute Y, since it appears in no object of this context. Given a new issue o, the `partition` method computes, for every context $\mathbb{K}$ in the partition, the closure $\mathsf{o}''_{\mathbb{K}}$ of o over $\mathbb{K}$. The proposed components are those that appear in any of these closures; that is, we propose

$$\mathsf{candidates}(\mathsf{o}) = C \cap \bigcup \{\mathsf{o}''_{\mathbb{K}} \mid \mathbb{K} \text{ subcontext in the partition}, \mathsf{o}'_{\mathbb{K}} \neq \emptyset\} \tag{1}$$

as candidates for the observation, where $(\cdot)'_{\mathbb{K}}$ and $(\cdot)''_{\mathbb{K}}$ denote the derivation and double derivation operator in the corresponding context $\mathbb{K}$.

The score of each proposed component $x \in C$ is given by the proportion of subcontexts $\mathbb{K}$ in the partition such that $x \in \mathsf{o}''_{\mathbb{K}}$, i. e.

$$\mathsf{score}(x) := \frac{|\{\mathbb{K} \mid \mathbb{K} \text{ subcontext in the partition}, x \in \mathsf{o}''_{\mathbb{K}}\}|}{k}$$

where $k = \lceil \frac{|G|}{n} \rceil$ is the number of contexts in the partition.

The closure of the observation $\mathsf{o}_{\mathsf{exa}} = \{b, c\}$ over the subcontexts $\mathbb{K}_1$ and $\mathbb{K}_2$ is $\{a, b, c, X\}$. Thus, component X is proposed with score 1 (since it appears in the closure for all the subcontexts), and component Y is not proposed.

While the `partition` method behaves well in our scenario of software issues, as shown in the following section, one might still increase its accuracy by allowing more components to be suggested. The `partition-pp` method achieves this by considering the *proper premises* for the components in each subcontext, rather than a direct closure.

Definition 3 (proper premise)**.** Given a set of attributes $B \subseteq M$, let

$$B^\bullet := B'' \setminus (B \cup \bigcup_{S \subsetneq B} S'').$$

B is a *proper premise* if $B^\bullet \neq \emptyset$. It is a *proper premise of* $m \in M$ if $m \in B^\bullet$.

The idea of considering proper premises arises from the existence of the partition $M = F \cup C$ of the attribute set. More precisely, we are interested in implicational dependencies "from F to C," i. e. implications $A \to B$ satisfying $A \subseteq F$ and $B \subseteq C$. Then sets $\mathcal{L}$ of implications of this type are *iteration-free*, i. e. the computation of closures $\mathcal{L}(\bar{F})$ of sets $\bar{F} \subseteq F$ can be achieved by

$$\mathcal{L}(\bar{F}) = \bar{F} \cup \bigcup \{B \mid (A \to B) \in \mathcal{L}, A \subseteq \bar{F}\}.$$

In other words, the computation given by the right-hand side of this equation does not need to be iterated to compute $\mathcal{L}(\bar{F})$ [5].

We now want to compute bases of this type of implications for each subcontext in `partition` and to use them instead of $(\cdot)''$. Of course, one would like to have such a set to be as small as possible, and indeed proper premises provide a way to obtain such a base. In other words, the set

$$\{B \to B^\bullet \cap C \mid B \subseteq F \text{ is a proper premise for some element in } C\}$$

is a minimal iteration-free base for all implications from F to C [2, 5]. This motivates the use of proper premises. Note that proper premises allow for interesting optimizations with respect to their computation [13].

We apply this idea as follows: for each subcontext in the partition of $\mathbb{K}_{\mathsf{reports}}$, the `partition-pp` method computes the proper premises of the components appearing in it. We only include those proper premises which have positive support withing this subcontext. For each such proper premise B for a component x, we collect the implication $B \to \{x\}$ into a set $\mathcal{L}$. Responsible components are then proposed by finding all collected implications $(B \to \{x\}) \in \mathcal{L}$ such that $B \subseteq \mathsf{o}$, and suggesting their associated components x. The score of suggesting x is given by the maximal confidence of an implication $(B \to \{x\}) \in \mathcal{L}$ such that $B \subseteq \mathsf{o}$, i. e.

$$\mathsf{score}(x) = \max\{\mathsf{conf}(B \to \{x\}) \mid (B \to \{x\}) \in \mathcal{L}, B \subseteq \mathsf{o}\}$$

where the confidence is computed in $\mathbb{K}_{\mathsf{reports}}$.

The proper premises for X in the context $\mathbb{K}_1$ are $\{a\}, \{b\}$, and $\{c\}$. In $\mathbb{K}_2$ there are no proper premises for Y and the only proper premise for X is $\{c\}$. The confidence of the implications $\{a\} \to \{X\}, \{b\} \to \{X\}$, and $\{c\} \to \{X\}$ over $\mathbb{K}_{\mathsf{exa}}$ is $\frac{3}{4}$, $\frac{2}{4}$, and 1, respectively. Thus, given our observation $\mathsf{o}_{\mathsf{exa}}$, only the component X is suggested with score 1, due to the implication $\{c\} \to \{X\}$.

We can expect `partition-pp` to return more candidates than `partition`, which is also confirmed by our experiments. This is due to the following reason: in (1), a candidate set $\mathsf{o}''_{\mathbb{K}}$ is excluded when $\mathsf{o}'_{\mathbb{K}} = \emptyset$, i. e. if no object in the subcontext has all the features in o. That is, we always consider the whole set o in every such subcontext. However, there might still exist a subset $\mathsf{p} \subseteq \mathsf{o}$ which meaningfully entails responsible components, in the sense that

$\mathsf{p}'_{\mathbb{K}} \neq \emptyset$ and $\mathsf{p}_{\mathbb{K}} \neq \mathsf{p}''_{\mathbb{K}}$. Those sets p are ignored in `partition`, but not in `partition-pp`: If $x \in \mathsf{p}''_{\mathbb{K}}$, then there exists a proper premise as subset p for x with positive support, and thus `partition-pp` proposes x as a candidate.

4. Results and Discussions

We implemented all methods described above on `conexp-clj`, a general-purpose library for formal concept analysis,[1] and applied them to data describing software issues, collected by a large German software company, considered as a multivalued context. The original data had six features that received several different manifestations. We scaled this context nominally, resulting in a formal context of size 2951×2973 with incidence density of roughly 0.002. We then conducted the following experiments to measure the quality of these methods with respect to classifying bug reports: for varying $n \in \mathbb{N}_+$, we randomly chose a subcontext with n objects, removing all attributes which have empty extent in the corresponding subcontext. Then $\lfloor 0.9 \cdot n \rfloor$ of these items were used to *train* the methods; i. e. formed the context of reports $\mathbb{K}_{\mathsf{reports}}$, and the remaining $\lceil 0.1 \cdot n \rceil$ data items were used to *test* them. A test consisted in classifying the set of features of the data items, and comparing the proposed components with the known responsible component. For each fixed n, the whole procedure was repeated five times; 5 different, randomly chosen subcontexts were considered. We recorded the averages of all the values measured during each of these five executions.

To evaluate the testing data, and obtain a better evaluation of our proposed methods, we also implemented a *random* classifier. This method simply proposes a randomly chosen proportion of all the available components. The number of proposed components is determined by an input parameter. This allows us to determine whether the components are uniformly represented in the data, and avoid giving special importance to over-represented components.

We tested the methods in two steps, when applicable: first, every method proposes a set of components as being responsible for the issue. This test is *positive* if and only if the original responsible component is among those proposed. We also measured the mean percentage of proposed components among all components in the data, to discern methods that yield positive answers simply because they propose a large amount of components, from those that yield more informative answers. Most of our methods also graded the proposed components. For those methods a test is *correct* if and only if the original responsible component is among the top-rated ones. Again, we also measure the mean percentage of the top-rated components.

[1] `http://github.com/exot/conexp-clj`

TABLE 4. Experimental Results

Method	n	train (ms)	test (ms)	positive	proposed	correct	proposed
`random`(0.2)	1000	7.25	26.32	20.81%	20.00%	–	–
`random`(0.2)	2000	15.12	24.87	19.09%	19.72%	–	–
`random`(0.5)	1000	4.85	16.64	51.38%	50.00%	–	–
`random`(0.5)	2000	22.63	30.65	51.16%	49.82%	–	–
`random`(0.9)	1000	12.58	26.21	90.50%	89.61%	–	–
`random`(0.9)	2000	13.37	33.40	87.00%	89.68%	–	–
`new-incident`	1000	0.02	19856.95	36.00%	3.11%	–	–
`new-incident`	2000	0.02	59371.25	44.50%	3.00%	–	–
`subcontext`(0.05)	750	3181598.62	11.98	69.33%	28.15%	30.67%	0.55%
`subcontext`(0.05)	1000	2841258.02	14.10	73.00%	29.94%	51.00%	0.54%
`subcontext`(0.01)	750	3355400.12	12.65	73.33%	25.48%	37.33%	0.50%
`subcontext`(0.01)	1000	2923139.74	15.09	72.00%	27.93%	41.00%	0.50%
`can+lux`(0.01,0.7)	1000	1375682.73	113.01	9.00%	0.05%	9.00%	0.05%
`can+lux`(0.01,0.7)	2000	3260189.07	219.74	8.50%	0.03%	8.50%	0.03%
`can+lux`(0.01,0.9)	1000	1359721.46	148.44	14.00%	0.07%	14.00%	0.07%
`can+lux`(0.01,0.9)	2000	3378045.62	199.46	7.50%	0.03%	7.50%	0.03%
`can+lux`(0.05,0.7)	1000	310803.29	0.24	0.00%	0.00%	0.00%	0.00%
`can+lux`(0.05,0.7)	2000	724341.14	0.13	0.00%	0.00%	0.00%	0.00%
`can+lux`(0.05,0.9)	1000	340270.58	119.62	5.00%	0.03%	5.00%	0.03%
`can+lux`(0.05,0.9)	2000	787725.99	0.09	0.00%	0.00%	0.00%	0.00%
`partition`(3)	1000	18961.75	193.25	33.94%	0.84%	30.00%	0.23%
`partition`(3)	2000	73898.59	519.63	49.13%	0.69%	47.00%	0.19%
`partition`(10)	1000	6161.62	109.95	34.00%	0.21%	34.00%	0.21%
`partition`(10)	2000	23895.38	268.83	46.00%	0.34%	45.00%	0.19%
`partition`(15)	1000	4943.22	109.95	36.00%	0.23%	34.06%	0.17%
`partition`(15)	2000	16468.57	245.98	41.69%	0.27%	40.56%	0.16%
`partition`(30)	1000	2488.06	94.70	40.00%	0.23%	40.00%	0.20%
`partition`(30)	2000	8529.07	218.02	44.50%	0.25%	43.50%	0.18%
`partition-pp`(3)	1000	89773.62	83.68	78.19%	31.12%	64.00%	0.50%
`partition-pp`(3)	2000	418524.89	217.82	88.38%	37.96%	71.00%	0.39%
`partition-pp`(10)	1000	142692.24	63.23	77.38%	7.32%	68.00%	0.52%
`partition-pp`(10)	2000	498504.77	151.32	82.19%	10.79%	72.22%	0.43%
`partition-pp`(12)	1000	182192.81	57.21	78.63%	7.30%	69.69%	0.49%
`partition-pp`(12)	2000	491516.69	120.77	81.66%	9.29%	70.84%	0.39%
`partition-pp`(15)	1000	267326.75	53.75	76.88%	7.08%	61.31%	0.51%
`partition-pp`(15)	2000	630232.06	109.49	80.91%	7.56%	70.91%	0.38%
`partition-pp`(30)	1000	1083661.64	38.19	66.63%	3.62%	59.00%	0.44%
`partition-pp`(30)	2000	2201360.65	85.13	80.94%	4.16%	71.56%	0.36%

The results are shown in Table 4. This table includes the training and testing times, which should be considered with care: the experiments were conducted in parallel on a 24 core machine, and the times measured are the overall execution times, not the ones per thread. Thus, the actual computation times could be lower than the ones stated in the table. However, these numbers still give a feeling on how these methods perform. Also note that we applied a timeout of 5 hours for each experiment, including repetitions.

From the experimental results we first see that the random classifiers behave as expected: if we choose randomly 20% of all components we have, then roughly 20% of the tests are positive; that is, the responsible component is among those proposed. The same is true for 50% and 90%. Thus, our data behaves mostly like random data with respect to our classification task. With respect to this random selection, even our simple approach `new-incident` performs much better: for $n = 1000$, only around 3% of the components are proposed while 36% of all tests were positive. This performance increases for $n = 2000$. However, while the training time is negligible (there is no training phase), the testing time is quite high, and increases with the size of the data. This may render this approach difficult to apply in realistic scenarios, where the classification time is the real bottleneck. Fortunately, only the `new-incident` method has such long testing times. In all the following approaches, the testing time is negligible. However, the price to pay are rather huge training times; sometimes even larger than the timeout used. On the other hand, in comparison to testing, training is conducted rarely, which means that huge training times can still be acceptable in practical applications.

The first method in this category is `subcontext`, which we applied with parameters 0.05 and 0.01 to our data. We see that the rate of positive tests is quite high, but also the percentage of components proposed. On the other hand, the scoring function provides a good method for further reducing the set of proposed components: only one out of each 200 components is rated with the highest score, and the correct answer is still provided in roughly half of the cases. However, the training times for this method were the largest among all the approaches we tested, by a broad margin. As an overall comparison with other methods, we conclude that the `subcontext` method is not the best suited for achieving a convincing classification.

The approach `can+lux`, which combines the canonical and Luxenburger's bases, performs even worse, much to our surprise: although the proportion between the number of proposed components and positive and correct tests is comparatively good, the latter is too low to be of any use for classification. Moreover, the rating provided by this method yields no improvement over the unrated classification. As the percentage of proposed components is almost the same, we can conclude that most components receive the same (highest) score. This behavior is not necessarily an intrinsic problem of the method, but could be attributed to a faulty choice of the scoring function. Notice, however, that the method proposes in average *less* than one responsible component. Thus, the same behavior would be observed, regardless of the scoring function.

This picture changes drastically when we come to the approaches based on partitioning the training data into smaller subcontexts. For `partition` we not only achieved rather high positive rates, but the number of proposed

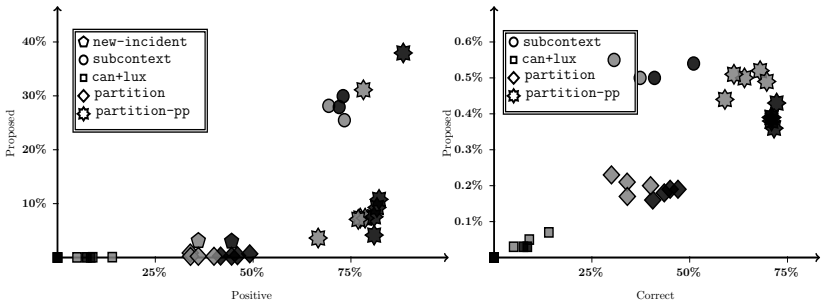


FIGURE 1. Positive (left) and correct (right) vs. proposed components

components was radically reduced. Interestingly, the rating provided by this method also behaves well; it keeps high rates of correct tests but reduces the number of proposed components. This is especially true if the partitioned contexts are very small (e. g. have 3 objects), but is also observable for larger contexts. Finally, the training times are practically irrelevant, and should scale well for even larger data sets. Notice that the training time depends linearly on the number of objects in the training data; if we additionally want to restrict to only the highest-ranking components, then the training time becomes $\mathcal{O}(n \log n)$ since an additional sorting step is needed.

While `partition` behaves relatively well, the proportion of positive tests remains below 50%. It would clearly be nicer to increase this number, even if the rate of proposed components increases. This is achieved by introducing implicational dependencies as in `partition-pp`, where both the positive and correct rates are increased. The cost of this improvement is to propose more components in both cases, but the ratios between proposed and positive or correct rates are still very good. What is very surprising, though, is that the rating provided by this approach is very effective, reducing the number of proposed components by factors of 10 or more while keeping high rates of correct tests. This is especially true for $n = 2000$; one can conjecture that this improves for larger training sets. Moreover, we can also see that the larger the subcontexts we consider in our partition, the smaller the sets of proposed components are. However, we have to pay for this with an increase in the training time, which may or may not be relevant in practice. In particular, this method proposes in average less than six top-rated components, and it might not be worth spending resources trying to reduce this number further.

The results of Table 4 are further depicted in Figure 1. The horizontal axis corresponds to the percentage of positively or correctly classified tests, respectively, while the vertical axis shows the percentage of suggested components. The ideal situation is an element in the lower-right corner: a high percentage of success, while suggesting only a few candidates. In the plots, different methods are depicted by different node shapes, while the shade of

gray expresses the size of the training set: a darker shade means a larger set. As it can be seen, the nodes sharing the same figure and the same shade of gray form natural clusters in these plots. This suggests that the quality of the results depends mainly on the method chosen, and the size of training set, while the parameters used in the specific method are not that relevant. The best results were obtained by `partition-pp` with a training set of size 2000. This corresponds to the cluster of nodes depicted by ✹ in the plots. It can be easily seen that this method indeed showcases the best behavior. The only exception is the case where partitions have size three, which is the single node in the upper-right corner of the left plot: it was the most successful w.r.t. positive classification, but suggested over a third of all available components.

These experimental results suggest that our initial idea of using implicational dependencies between attributes to classify bug reports is reasonable, but only if considered "locally" as in `partition` and `partition-pp`. If those dependencies are considered in the overall training data, then the resulting classification fails miserably (see `can+lux`). The partitions used in `partition` and `partition-pp` should not contain too large, nor too small, contexts.

For putting these methods into practice, we can also think of a combined approach of `partition` and `partition-pp`: the former one has an acceptable performance and suggests only very few components. Therefore, considering those components may be a good starting point. If the responsible component is not among those proposed by `partition`, one can consider those proposed by `partition-pp`, which may be more (especially if not rated), but which are also more likely to contain the real cause of the issue. Also different sizes of the partition are imaginable, increasing the performance of the classification but also the number of proposed components. If all fails, one has to fall back to manual classification. However, this last resource is needed only sporadically.

5. Conclusions

Our goal was to analyze whether FCA tools can be useful for classifying software issues according to their responsible component. Contrary to standard machine learning techniques, FCA methods provide logical implications between the symptoms (features of the bug) and the causes (the responsible component). These implications can be understood by users, and provide more detailed information of the software system itself. The use of association rules to detect faults and vulnerabilities in software systems has been studied previously [3, 4, 12]. The main difference with this paper is that we study and compare different approaches for handling erroneous and incomplete information, and detected empirically which is best suited for our scenario.

We tried several approaches, all based in ideas developed in FCA. Each of the methods was inspired by different approaches towards the problem. One of the important issues was how to deal with potential errors, incomplete knowledge, and change of the software structure over time. Surprisingly to us, the obvious idea of using Luxenburger's base to handle uncommon exceptions yielded relatively bad results: the responsible component was usually not proposed, regardless of the chosen minimal support and confidence.

The method that behaved best in our scenario was to compute a base of proper premises over a partition of the historical records, together with a scoring function for the proposed components. This method behaves very well from partitions of size 3 up to 30, yielding the right answer in over two-thirds of the cases, while proposing less than 0.5% of the available components. This method also scales well: whenever new historical records are added, only the proper premises over a partition of the new cases need to be computed. All previous records remain unchanged. Moreover, it is easy to get rid of old historical records, by simply deleting their corresponding partitions.

In general, our experiments show that it is feasible to classify objects from large historical records using FCA, provided that training can be done off-line. While training in the `partition-pp` method could take more than 10 minutes, in a context of 1000 objects, the classification time was almost instantaneous, taking less than 100ms. For our software issue scenario, these conditions are satisfactory: new issues would be entered to the training data sporadically, and training may take place over-night. However, lower classification times, with higher success rates and small sets of candidate components, translate into faster repair of software bugs.

We have not compared our approach with existing classification methods from machine learning and other areas. Since we obtained promising results, we will make such comparison in the future. Our implementation is prototypical, and requires further optimization for industrial-strength use. Studying some applicable optimization techniques will also be a focus of future work.

References

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. "Mining Association Rules between Sets of Items in Large Databases". In: *Proc. ACM SIGMOD Int. Conf. on Management of Data*. 1993, pp. 207–216.

[2] Karell Bertet and Bernard Monjardet. "The multiple facets of the canonical direct unit implicational basis". In: *Theoretical Computer Science* 411.22-24 (2010), pp. 2155–2166.

[3] Peggy Cellier. “Formal concept analysis applied to fault localization”. In: *Companion Volume of the 30th International Conference on Software Engineering*. (Leipzig, Germany). ACM, 2008, pp. 991–994.

[4] Peggy Cellier et al. “Formal Concept Analysis Enhances Fault Localization in Software”. In: *Proc. ICFCA 2008*. (Montreal, Canada). Vol. 4933. LNCS. Springer, 2008, pp. 273–288.

[5] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Berlin-Heidelberg: Springer, 1999.

[6] Gaeul Jeong, Sunghun Kim, and Thomas Zimmermann. “Improving bug triage with bug tossing graphs”. In: *Proc. ACM SIGSOFT Int. Symp. on Found. of Software Eng.* ACM, 2009, pp. 111–120.

[7] Sergei O. Kuznetsov. “Complexity of learning in concept lattices from positive and negative examples”. In: *Discrete Applied Mathematics* 142.1-3 (2004), pp. 111–125.

[8] Sergei O. Kuznetsov. “Machine Learning and Formal Concept Analysis”. In: *Proc. ICFCA 2004*. Vol. 2961. LNCS. Springer, 2004, pp. 287–312.

[9] Michael Luxenburger. “Implications partielles dans un contexte”. In: *Mathématiques, Inform. et Sciences Humaines* 29.113 (1991), pp. 35–55.

[10] Michael Luxenburger. “Implikationen, Abhngigkeiten und Galois-Abbildungen”. German. PhD thesis. TH Darmstadt, 1993.

[11] Thomas M. Mitchell. *Machine Learning*. 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[12] Stephan Neuhaus and Thomas Zimmermann. “The Beauty and the Beast: Vulnerabilities in Red Hats Packages”. In: *Proc. 2009 USENIX Annual Technical Conference*. 2009.

[13] Uwe Ryssel, Felix Distel, and Daniel Borchmann. “Fast algorithms for implication bases and attribute exploration using proper premises”. In: *Annals of Math. and Artif. Intel.* Special Issue 65 (2013), pp. 1–29.

[14] Gerd Stumme et al. “Intelligent Structuring and Reducing of Association Rules with Formal Concept Analysis”. In: *Proc. KI 2001*. (Vienna, Austria). Vol. 2174. LNCS. Springer, 2001, pp. 335–350.

Theoretical Computer Science, TU Dresden, Germany
E-mail address: daniel.borchmann@mailbox.tu-dresden.de

TCS, TU Dresden, Germany. Center for Advancing Electronics Dresden
E-mail address: penaloza@tcs.inf.tu-dresden.de

SAP, Germany
E-mail address: wenqian.wang@sap.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Special Issue 2, 2014
ICFCA 2014: 12th International Conference on Formal Concept Analysis, Cluj-Napoca, June 10-13, 2014

STABILITY-BASED FILTERING FOR ONTOLOGY RESTRUCTURING

SCHAHRAZED FENNOUH, ROGER NKAMBOU, PETKO VALTCHEV, AND MOHAMED ROUANE-HACENE

Abstract. Assessing the relevance of concepts extracted from data is an important step in the knowledge discovery process. We address this issue in a specific outfit, i.e., the discovery of new ontological abstractions by relational concept analysis (RCA). In the context of RCA-based ontology restructuring, potentially relevant abstractions must be recognized among the formal concepts of the output lattice before integrating them into the restructured ontology. Thus, a key technical challenge is the design of effective relevance-based filtering methods. In our study, we examined a variety of relevance measures. Here, we focus on concept stability and discuss its usefulness in the light of the outcome from an experimental study involving several ontologies retrieved from the Web.

1. Introduction

An ontological model is like a database conceptual schema [8]: it provides the framework in which to fit the fine-grain knowledge about a particular domain or subject. Like most artifacts in information system development (conceptual models, design models, source code, etc.), an ontological model is prone to errors and design anomalies. Previous attempts at detecting and, possibly, correcting such anomalies yielded a variety of *restructuring* approaches. Intuitively, a restructuring process aims at improving the quality of an ontology, which further increases usability and eases maintenance [3]. Technically speaking, ontology restructuring reshuffles its current structure into a new one, better organized and more complete. It thus refers to: (1) correction and reorganization of knowledge contained in the initial conceptual model, and (2) the discovery of missing knowledge pieces and their integration into the improved structure [19].

Received by the editors: March 27, 2014.

2010 *Mathematics Subject Classification.* 03G10, 68T30.

1998 *CR Categories and Descriptors.* I.2.4 [**K**nowledge Representation Formalisms and Methods]: Representations.

Key words and phrases. Filtering, Relevance evaluation, Concept stability, Ontology restructuring.

The problem has been addressed in the literature from a variety of standpoints [3, 9, 20]. However, there is no such thing as a well-established methodology covering the variety of restructuring steps and techniques. Even worse, none of the proposed solutions offers a holistic approach to the ontological structure. Instead, local changes are focused on, without insight on their impact on distant parts of the structure. In addition, existing methods are limited to problem detection and improvement, leaving the other crucial restructuring aspect, i.e., the discovery of missing knowledge chunks, uncovered.

Following the success of FCA-based restructuring methods in software re-engineering [6], we propose a similar approach for ontologies. Indeed, FCA provides the formal framework necessary to support a truly holistic approach towards restructuring while simultaneously propping up new abstractions through factoring out shared descriptions. Moreover, due to the complex relational information comprised in a typical ontological model, we propose to use the Relational Concept Analysis (RCA) extension. Yet the mathematical strength of FCA and the expressiveness of RCA come with a price: A key challenge to face here is the complexity of the resulting relational lattices. A standard way out in such cases is the design of effective filtering methods that help spot and then remove the spurious concepts that abound in the output lattice. Thus, our overall goal is the design of appropriate decision criteria for relational concepts or, alternatively, means to assess their relevance.

Selecting relevant concepts within a lattice is knowingly a delicate task for which few generic hints are available. Indeed, relevance is a contextual and subjective property. Therefore, fully automated filtering methods rely on structural properties that are easier to measure. In our restructuring context, however, the input ontology, albeit of a flawed structure, constitutes a rich source of domain knowledge to explore in the design of "semantic" relevance measures. Yet at this stage, we chose to keep to a purely structural approach and ignore the ontology. Thus, we adapted concept *stability* as defined in [10] to our RCA framework. The present paper is a report on an experimental evaluation of the resulting measure's usefulness.

The remainder of the paper is organized as follows; we start, in section 2, by presenting our RCA-based approach for ontology restructuring; we recall the basic notions of FCA/RCA framework and present the problem of lattice filtering. Section 3 is devoted to the definition of the notion of stability and its projection in an ontological context; and then the proposal of a simple filtering heuristic based on stability. We explain, in section 4, our experimental framework followed by our experimental study including the discussion of our results. Section 5 provides an overview of related work. Finally, section 6 provides concluding remarks with an outlook of future work.

TABLE 1. Key differences with standard FCA notations.

Not.	Description	Not.	Description
O	The set of formal objects	$\mathcal{C}_K^o$	The family of extents of a context K
A	The set of formal attributes	$\mathcal{C}_K^a$	The family of intents of K
$\mathcal{C}_K$	The set of formal concepts of K	$\mathcal{L}_K$	The concept lattice of K

Classes	author	reviewer	paper	report	name	affiliation	email	rank	title	contents	comments	login	passw
Author	×				×	×		×					
Paper			×						×	×			
Reviewer		×			×	×	×					×	×
Report				×							×		

Properties	'write'	'compose'	'rate'
Write	×		
Compose		×	
Rate			×

FIGURE 1. Contexts K_1 (of *Classes*) and K_2 (of *Object properties*) of CMS ontology.

2. Ontology Restructuring Using RCA

FCA has been successfully applied as a formal framework for the design/restructuring of class hierarchies in OO languages [4, 6] and of conceptual hierarchies in knowledge representation [18, 12]. Below, we first recall basic notions from FCA and RCA, then present the overall restructuring method and finally state the filtering problem for concept lattices output by RCA.

2.1. **RCA basics.** The notations we use in the remainder of this paper slightly diverge from the standard ones. The important differences are summarized in Table 1.

The aim of RCA [17] is to discover formal concepts on top of multiple object sets described by both proper attributes and links. In RCA, data encoding is done through a structure called *Relational Context Family* (RCF). RCF is a pair **(K,R)**, such that: $\mathbf{K} = \{K_i\}_{i=1,...,n}$ a set of contexts $K_i = (O_i, A_i, I_i)$, each representing an object species, and $\mathbf{R} = \{r_k\}_{k=1,...,m}$ a set of relations r_k where $r_k \subseteq O_{i_1}$ x O_{i_2} for some $i_1, i_2 \in \{1, ..., n\}$, with O_{i_1} (domain of r_k) and O_{i_2} (range of r_k) are the object sets of the contexts K_{i_1} and K_{i_2}, respectively. Fig. 1 shows two contexts K_1 and K_2 of a *Conference Management System* (CMS) ontology representing two classes of ontological entities, *concepts* and *object properties*, respectively (see Fig. 2 for the corresponding concept lattices $\mathcal{L}_1$ and $\mathcal{L}_2$).

Existing links between *concepts* and *object properties* are represented by four relations (*source, target, domain and range*) as showed in Fig. 3. For instance, the *domain* relation models the relationship between a property and

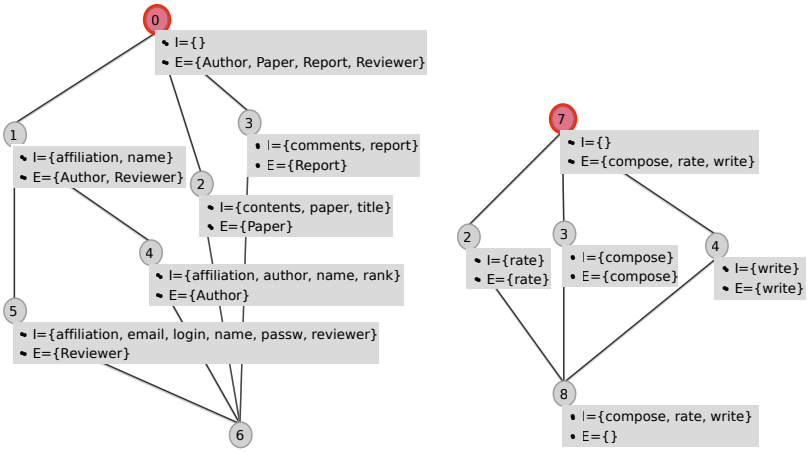


FIGURE 2. The concept lattices of the contexts K_1 (left) and K_2 (right).

source	Write	Compose	Rate
Author	×		
Paper			
Reviewer		×	
Report			×

target	Write	Compose	Rate
Author			
Paper	×		×
Reviewer			
Report		×	

domain	Author	Paper	Reviewer	Report
Write	×			
Compose			×	
Rate				×

range	Author	Paper	Reviewer	Report
Write		×		
Compose				×
Rate		×		

FIGURE 3. Relations of CMS Relational Context Family.

the own class and the *source* relation expresses the semantic of "is the domain of".

To deal with the relational structure of an RCF, a mechanism called *scaling* transforms inter-object links into descriptors for formal objects. As a result, new attributes called *relational* are added to the attribute sets A_i from the RCF. Thus, for $K_j = (O_j, A_j, I_j) \in \mathbf{K}$, A_j is extended with attributes $a_{r,c}$ where r is a relation from $\mathbf{R}$ such that *dom(r)* $\subseteq O_j$, and c is a concept over the objects from *ran(r)*. Furthermore, such attributes involve a quantifying operator (universal, existential, existential with cardinality restriction, etc.). In our CMS case, all such attributes are assumed to refer to an existential quantifier. For instance, in Fig. 4 (left hand side), the attribute source:c5 of the concept $c_{\#1}$ is to be read as: *For each class* o *in the extent of* $c_{\#1}$, *exists an object property* p *in the extent of* $c_{\#5}$ *(*right hand side*) such that* o *is the source of* p.

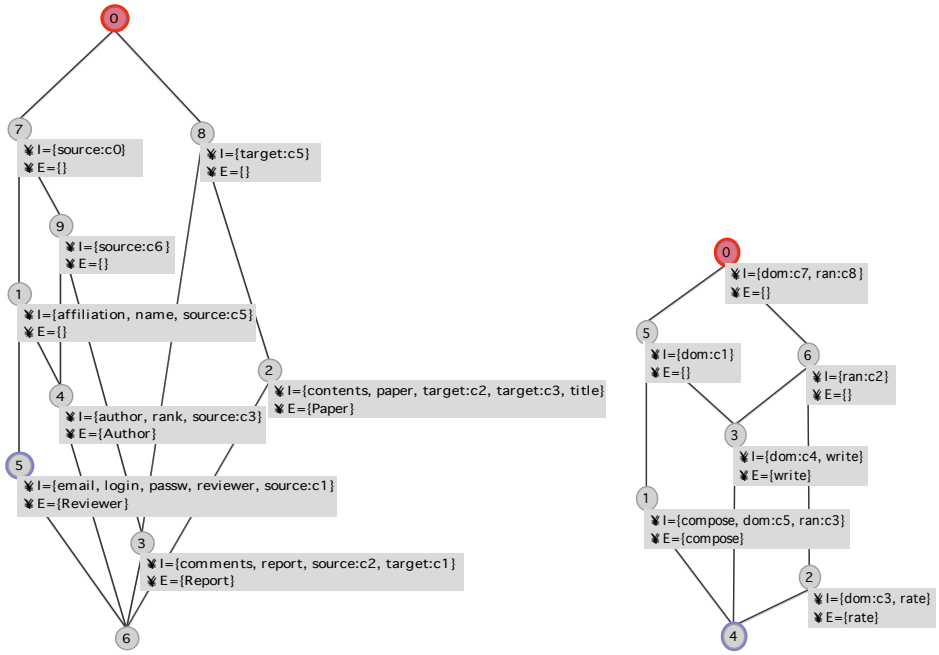


FIGURE 4. The fixed-point lattices of CMS ontology.

The overall process of RCA follows a Multi-FCA method which allows to construct a set of lattices, called *Relational Lattice Family* (RLF), one per relational context. The RLF is defined as a set of concepts that jointly reflect all attributes and links shared by the objects of the RCF. The logic of this analysis method is iterative one: Whenever the contexts of RCF are extended, their corresponding lattices expand as well. At the initialization step, each context K_i^0 is obtained from K_i by applying a conceptual scaling to the many-valued attributes in K_i and then the lattices $\mathcal{L}_i^0$ are constructed. At the step p and for each relation $r_k \subseteq O_i \times O_j$, the lattice $\mathcal{L}_j^{p-1}$ of the range context is used to extend the domain context K_i^p and then update the lattice of the domain context $\mathcal{L}_i^p$. The process ends whenever at two subsequent steps all the pairs of corresponding lattices $\mathcal{L}_i^n$ and $\mathcal{L}_i^{n+1}$ are isomorphic (i.e., the fix point solution is reached). For instance, the fixed-point lattices of the CMS example are depicted in Fig. 4 : within the property lattice (see the right of the figure), the concept $c_{\#5}$ summarizes the commonalities of write and compose which amount to a shared domain class, represented by the concept $c_{\#1}$ from the class lattice (see the left hand side of the figure), that is the super-class of Author and Reviewer.

2.2. **Restructuring Approach.** Our restructuring approach follows five steps: alignment, encoding, analysis, filtering and reverse encoding. The *alignment* step compares the elements of the initial ontology to identify similarities. This will eliminate redundancies and avoid duplication in the codes of similar ontological elements. In *encoding* step, the initial ontology will be transformed into a unique Relational Context Family (RCF) where each context represents a class of ontology metamodel elements (e.g. concepts and roles). Existing links in the metamodel between these two sorts of elements are represented by cross-context relations (e.g. source, target, domain and range). The *analysis* step consists of construction of the initial lattices (concepts and roles lattices) with FCA, then translates the cross-context relations into relational attributes following "relational scaling" mechanism. Next, the final concept lattices are constructed according to RCA iterative process converging towards a global fix-point of the analysis.

Please notice that the final lattices provide the support for the reorganization of the initial ontology. The *filtering* phase will filter these lattices by pruning of uninteresting and spurious formal concepts. The way to proceed at that stage can be summarized as follows:

(1) *identifying the ontological concepts of the domain (from the initial ontology)*,
(2) *selecting the relevant new abstractions.*

The final step is the *reverse encoding* that consists to generate semi-automatically (with the participation of the expert) the restructured ontology model. The idea is to translate the formal concepts considered as relevant in the filtered lattices into ontological elements in OWL format.

2.3. **Filtering of Concept Lattices.** While providing a strong mathematical background, FCA rises a serious issue that is rooted the overly large-sized lattices which typically contain many spurious concepts. RCA tends to generate even larger lattices since it deals with richer data structures. In our context, a *formal concept* will represent an *ontological concept* (or class) where the intent and the extent correspond, respectively, to the set of the properties of the latter and the set of its sub-concepts.

In the final lattice produced by RCA, each formal concept represents a candidate likely to be selected as an ontological concept by an expert if deemed relevant (or otherwise if irrelevant). Thus, two related questions can be asked:

- *What is the ontological value of a formal concept?*
- *How to recognize among the large number of candidates the relevant ones?*

In studies from the literature, such as [18, 12] aiming at learning or merging of ontologies with FCA, there is no suggestion on an evaluation method of formal concept quality. Most of them focus on expert-based validation to create the target ontology from the final concept lattices. We tried to address this issue in our context of ontology restructuring using RCA. Our objective is to propose effective metrics to evaluate the relevance of formal concepts which will constitute the restructured ontology.

To that end, we took some inspiration from: (1) the principles and requirements that must be respected by an ontological model [7]; (2) the work on the evaluation of ontologies, including those which consider ontology as a graph and try to detect its structural and semantic characteristics [1]; and (3) metrics for lattices pruning [11, 14]. We chose four metrics inclusive two structural ones: *Density* indicates the usefulness of a concept in terms of the additional information it provides w.r.t. its neighborhood. *Stability* measures the dependency of a concept upon single objects/attributes of its. The remaining two are semantics-based: *Semantic Similarity between children concepts* reflects the semantic correctness of a concept, i.e., to what extent it subsumes similar sub-concepts. *Semantic Similarity with the user center of interest* assess the level to which the concept is rooted in the important concepts from the initial ontology (in terms of direct or indirect links).

In the following sections, we will focus on the stability measure and discuss correlation between the stability of a formal concept and the relevance of its translation as an ontological concept.

3. Filtering of Relevant Concepts based on Stability

3.1. **Stability.** The idea of stability has been used to assess plausibility of hypotheses of different kinds. In this line of thought, [10] has introduced the realization of the idea of stability of hypotheses based on similarity of object descriptions, and has extended it to the formal concepts. Accordingly, in this paper, we will study the utility of this measure for the evaluation of the relevance of formal concepts by taking into account the two following points: (1) Richer structures; and (2) Ontological context. In our work, we use the definition of stability as follows :

Definition 1. Let *K=(O,A,I)* be a formal context and *(X,Y)* be a formal concept of *K*. The stability index, σ, of *(X,Y)* is defined as follows :

$$\sigma(X,Y) = \frac{|\{Z \subseteq X | Z^{'} = X^{'} = Y\}|}{2^{|X|}} \tag{1}$$

With : X, the set of objects (extent) and Y, the set of attributes (intent). The stability index of a concept indicates how much the concept intent depends on particular objects of the extent. In other terms, the stability index represents the probability for a concept to preserve its intent even if some objects of its extent disappear. The idea behind stability is that a stable intent is probably "real" even if the description of some objects is "noisy".

3.2. **Stability in an Ontological Context.** Below, we project stability in our ontological context and propose an interpretation of the result. In other terms, the idea is to determine the ontological qualities/characteristics that can be represented by stability.

In an ontology or a taxonomy, an abstraction is basically a grouping of sub-classes that share some properties. Then, the set of shared properties constitutes the description of the abstraction. If we focus on relevance as a specific component of the overall concept quality, then successfully approximating that quality by stability amounts to showing that the following assumption holds:

Assumption 1. Given a class cl with sub-classes $C = \{c_1, ..., c_n\}$ and described by a set of properties $P = \{p_1, ..., p_m\}$ then, the bigger the number of subsets $X \subseteq C$, such that the members of X share exactly the set of properties P, the higher the relevance of cl.

In the following sections, we present an experimental study that put this assumption to test. We present, first, our heuristic of filtering which underlies the experiments.

3.3. **Stability-Based Filtering.** Our filtering method has the following overall structure:

Inputs:: Relational Lattice Family generated by the RCA-based restructuring method.

Outputs:: Set of formal concepts (those deemed relevant).

Body:: Computation of the evaluation metrics.

The following is a simple heuristic based only on the stability measure. The principle of this heuristic is illustrated by the following rules. Given a formal concept FC,

Rule 1:: Extent cardinality = 1 (object concept); Stability is invariably 0.5. The concept is assumed **relevant**.

Rule 2:: Extent cardinality = 2; Stability is either 0.25 (two sub-concepts) or 0.5 (single sub-concept). In this case the stability value is unrelated to the relevance, so the case is deemed **inconclusive**.

Rule 3:: Extent cardinality $\geqslant 3$; Stability value in the unit interval. A threshold criterion is applied as follows:

Rule 3.1:: If value $\geqslant 0.5$ the concept (a stable one[1]) is deemed **relevant**, otherwise, it is **irrelevant**.

4. Implementation and Validation Study

Below we present the experimental study aimed at testing the **Assumption 1**.

4.1. **Software Environment.** Our approach was implemented within the Inukhuk platform [16] which is a service-oriented infrastructure based on RCA with a set of tools for *Ontological Engineering*. Inukhuk includes services for ontology construction, modularization, merge and restructuring. Inukhuk is coupled with Galicia platform providing RCA services, as well as other platforms and APIs (e.g. Jena, Wordnet, Wikipedia, Gate, Align, SimPak).

An initial work [15] attempted to validate the RCA-based restructuring framework. Experiments have been carried out on medium-size ontologies which confirmed the satisfactory performances of the tool in terms of reorganization and identification of new abstractions. These showed the limits of the "naive" lattice filtering algorithm that was initially implemented and thus underscored the need for more effective filtering strategies.

Our filtering tool, called *RLFDistiller*, receives as input an RLF and outputs a set of relevant formal concepts. To date, three metrics are implemented: stability and density to evaluate the structural relevance, and semantic similarity with object concepts to evaluate the semantic relevance.

4.2. **Experiments.** An appropriate validation of the approach should follow an outline like: (1) Selection of a set of poorly designed ontologies; (2) Application of our restructuring tool to each of these ontologies and generation of relational lattices; (3) Evaluation by human experts of formal concepts representing the new discovered abstractions; (4) Application of our filtering tool on relational lattices; and (5) Correlation between filtering results and experts judgements.

However, due to the difficulties in obtaining poorly structured yet plausible ontologies and the scarcity of domain experts eager to collaborate in such experiments, we proceeded as follows:

(1) **Selection of a set of good quality ontologies** (complete and without redundancies) which will be considered as *reference ontologies*.
(2) **Focused perturbations of selected ontologies**. In order to generate poorly designed ontologies we introduce *redundancies* and *incompleteness*. Incompleteness comes from removing non leaf classes

[1]Please notice that the threshold of 0.5 is the one used in the literature.

from the class hierarchy. In doing that, we nevertheless preserve the properties –datatype and object ones– and property restrictions of the class that disappears: Whenever necessary, these are transferred to the sub-classes[2]. In this way, some redundancy may be generated. The resulting ontologies are called *degraded ontologies*. It is noteworthy that due to the way FCA-based restructuring works, all classes in a degraded ontology will appear in the *restructured ontology*.

(3) **Construction of relational lattices**. The restructuring tool is applied to each degraded ontology and relational lattices are generated. Such a lattice includes at least the following three categories of formal concepts representing, respectively: (1) initial concepts that are kept in the degraded ontology; (2) "missing abstractions" (removed concepts) rediscovered by the tool; and (3) new abstractions discovered by the tool (no equivalent in the reference ontology). In general, such abstractions could be deemed either relevant or not, by an expert. However, we intentionally chose complete ontologies so that the only relevant abstractions discovered by the tool correspond to concepts we removed.

(4) **Application of *RLFDistiller* on relational lattices**. The tool outputs a set of relevant formal concepts to become the *restructured ontology*.

(5) **Confrontation of each *restructured ontology* with its *reference ontology***. In order to measure the accuracy and completeness of our approach, we use the *precision* and *recall* measures as defined in Information Retrieval [13]. They require, in turn, the calculation of the following four sets:
   - **True Positives (TP)**: Concepts from the restructured ontology that have an equivalent in the reference ontology.
   - **False Positives (FP)**: Concepts from the restructured ontology that have no equivalent in the reference ontology.
   - **True Negatives (TN)**: Formal concepts from the relational lattice deemed irrelevant by the tool that have no equivalent ontological concept in the reference ontology.
   - **False Negatives (FN)**: Formal concepts from the relational lattice deemed irrelevant by the tool that have an equivalent ontological concept in the reference ontology.

[2]Obviously, this way to perturb ontologies ensures that all removed abstractions will be discovered by the RCA-based tool.

TABLE 2. Statistics on the *reference ontologies*.

Ontology name	Classes #	Object prop. #	Datatype prop. #	Individuals #
Cms	6	5	10	0
Travel	34	6	4	14
People	60	14	1	21
Tourism	76	26	27	111

TABLE 3. Examples of perturbations.

Ontology	Perturbation
Travel_degraded	1. Remove the *RuralArea* class and move its property *id_RuralArea* to the subclasses. 2. Remove the *Activity* class and move its properties *hasActivity* and *isOfferedAt* to the subclasses.
People_degraded	1. Remove *Company* class and transfer its property *id_company* to the subclasses. 2. Remove *Publication* class and transfer its properties *id_publication* and *reads* to the subclasses.

We applied the above protocol to a number of small/medium-sized ontologies whose size-oriented metrics are provided in Table 2. Table 3, in turn, exemplifies typical focused perturbations.

Now, in the analysis of the results generated by our tool, we followed a three-fold question as stated below. The goal was to verify to which extent:

(1) Concepts from the degraded ontology have high stability values ($\geqslant 0.5$).
(2) Removed abstractions are represented by formal concepts with high stability values ($\geqslant 0.5$).
(3) Other discovered abstractions (no equivalent in the reference ontology) have low stability values (< 0.5).

4.3. **Experimental Results.** The outcome of our experimental study are summarized in Table 4 which provides the respective cardinalities for the above four sets and for each ontology. It also cites the values of the quality metrics. Below, we illustrate the four cases.

- **TP :** In the relational lattice of the Travel ontology:
  - Concepts $c_{\#5}$ and $c_{\#23}$ (see Fig. 5) which represent two abstractions in the degraded ontology (Destination and UrbanArea, respectively) were deemed relevant (stability values of 0.91 and 0.68, respectively).

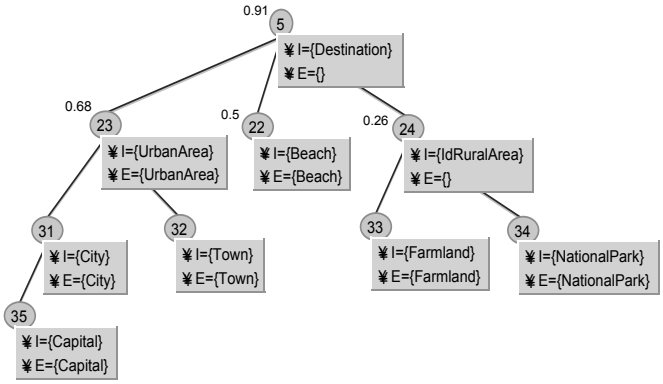


FIGURE 5. A part of concept lattice of Travel ontology.

  - Concept $c_{\#43}$ (see Fig. 6) representing the removed abstraction Activity emerged by factoring out shared links to properties; due to its high stability (0.68), it was labeled relevant by the tool.
- **TN :** In the Travel ontology again (see Fig. 6) concepts $c_{\#45}$, $c_{\#42}$ and $c_{\#46}$ represent newly discovered abstractions. In the lattice, they are also super-concepts of $c_{\#43}$ (Activity). For instance, $c_{\#45}$ groups Destination and Activity, hence it corresponds to an overly general notion. All three concepts are deemed irrelevant due to low stability (0.46, 0.48 and 0.49, respectively).
- **FP :** Concept $c_{\#66}$ from the lattice of the People ontology (see Fig. 7) has a stability of 0.75 and is therefore labeled relevant by the tool. However, the abstraction that it represents is too general so it doesn't exist in the reference ontology (grouping subclasses of Adult with subclasses of Publication).
- **FN :** Concept $c_{\#8}$ with the People ontology (see Fig. 7) represents the removed abstraction Publication that was indeed rediscovered. Thus it is a legitimate concept to keep in the restructured ontology. However, it was filtered out by the tool since its stability of 0.46 is below the threshold.

4.4. **Discussion.** From the above results, the following partial conclusions could be drawn: First, there is clearly no perfect correlation between stability of a concept and its relevance. In a slightly more negative tone, stability could not even be used to order concepts w.r.t relevance: Higher stability does not necessarily mean higher relevance.

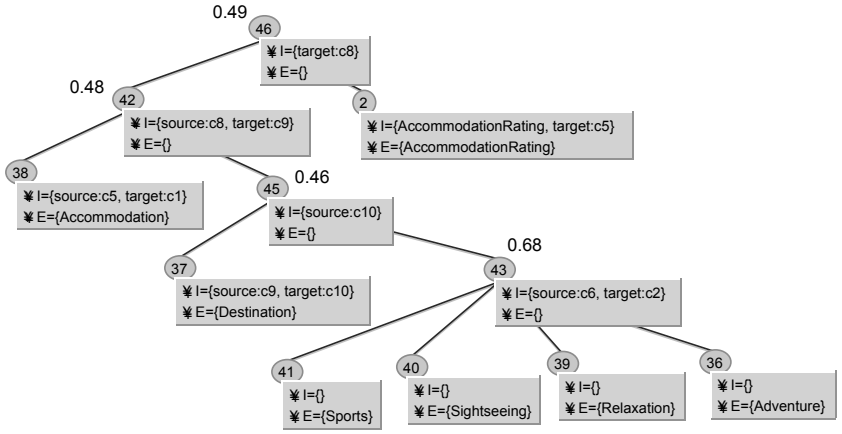


FIGURE 6. A part of concept lattice of Travel ontology.

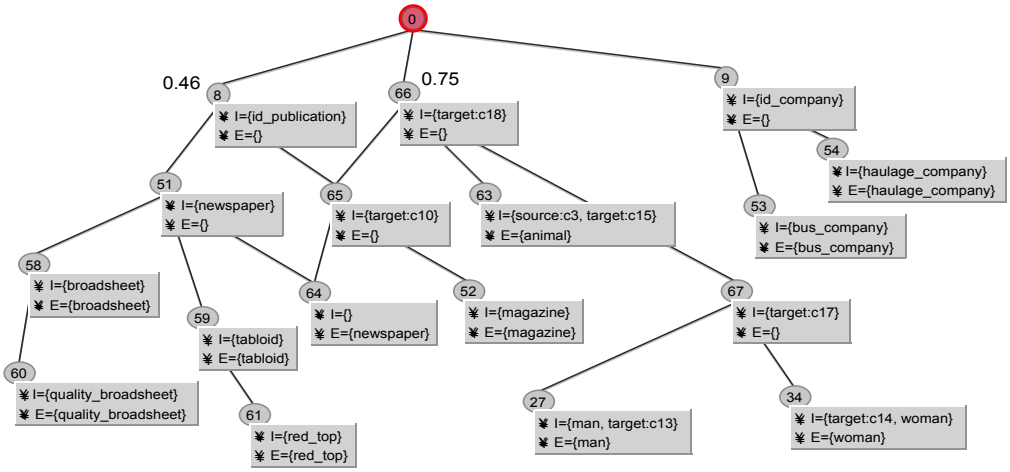


FIGURE 7. A part of concept lattice of People ontology.

On the positive side, 100% of the concepts in the degraded ontologies got high stability scores and therefore were deemed relevant by the tool. This speaks in favor of using stability as a component in the target filtering heuristic, possibly complemented by other measures.

TABLE 4. Final Results of experiments.

Ontology name	TP	FP	TN	FN	Recall	Precision	F-score
Cms	4	0	2	2	0.66	1	0.80
Travel	33	0	3	1	0.91	1	0.95
People	59	2	3	1	0.95	0.96	0.95
Tourism	75	9	21	1	0.78	0.89	0.83

Next, 37.5% of the rediscovered abstractions were labeled relevant[3]. While this rate might seem low, it is worth noting that another 50% got an *inconclusive* label. In theory, these could be retrieved by a different metric in the final filtering tool. Hence the real recognition rate for missing abstractions as to the current study is between 37.5% and 87.5%. However, in the assessment of our tool (see table 4), we took a conservative approach and we re-labeled all inconclusive cases as irrelevant.

Finally, spurious abstractions discovered by the RCA were filtered out by the tool with a 51.25% rate whereas another 24.39% were deemed inconclusive (again chances are they got stricken by another relevance metric). Thus, the overall pruning rate for irrelevant abstractions lays between 51.25% and 75.64%.

As a general conclusion, in the light of this first batch of experiments, concept stability seems to be a fair approximation of the relevance in an ontological context. Moreover, we tend to see the performances of our filtering tool as satisfactory and encouraging, in particular, w.r.t. to the improvements it brings to the complete ontology restructuring tool.

5. Related Work

FCA has been successfully applied to problems arising with class hierarchy design, maintenance and refinement [4, 6]. Moreover, several studies have used FCA as part of a process of ontology engineering. In [12], the authors have introduced a FCA-based methodology for the design of formal ontologies for product families. Terms describing the components in a product family along with the required properties are captured in a lexicon set and put into context. A class hierarchy is then derived from the concept lattice and exported into OWL. [18] have explored ontology construction by merging two existing ontologies provided with a corpus of textual documents. NLP techniques are used to capture the relationships between documents and classes from an ontology and organize them into a dedicated context. The two contexts are then merged and a pruned concept lattice is constructed which is

[3]Recall that the tool will invariably discover all of them

further transformed into a merged ontology by a human expert. The pruned concept lattice is computed with the algorithm TITANIC. This algorithm computes the formal concepts via their key sets (or minimal generators). A key set is a minimal description of a formal concept. These key sets are used, firstly, to indicate if the generated formal concept gives rise to a new concept in the target ontology or not. A concept is new if and only if it has no key sets of cardinality one. Secondly, the key sets of cardinality two or more can be used as generic names for new concepts and they indicate the arity of new relations.

The notion of stability has been used to prune concept lattices, notably, in the fields of social network analysis for dealing with communities. The goal in [11] is to select potentially relevant concepts based on their stability degrees. The method has been applied to large datasets from other domains as well. In [14], the authors apply FCA as a representation framework for the knowledge about the structure of a given knowledge community (based on shared vocabulary and topics of interests). Stability has been used here to prune the lattice so that only a sub-hierarchy thereof could be kept, comprising the most interesting concepts.

6. Conclusion and Future Work

Our ultimate goal is to improve the structural and semantic quality of an ontology. To that end, we study a restructuring approach based on a relational extension of FCA. Here, we focus on a crucial stage of the restructuring process, i.e., the filtering of the concepts from the output lattices and tackle the question of assessing their relevance. Relevance being contextual and subjective, we are studying various metrics to approximate it.

In this paper, we examine the stability measure and attempt an evaluation of its usefulness for our goals. We thus carried out a row of experiments in which we took an existing ontology, purposefully degraded its quality – structure and completeness–, run our restructuring tools, applied stability-based filtering, and finally compared the resulting set of concept deemed relevant to the initial ontology. The results of the experiments seem to reveal the following picture: While the experimental hypothesis of a good correlation between stability and relevance is not universally valid, which is hardly a surprise. Yet if we restrict the evaluation to formal concepts whose extents have *at least three* formal objects (i.e., sufficiently general), the correlation greatly improves.

The next step of the experimental study would be to put the apparent threshold of 50% that seems to work relatively well to test: Could this value be the manifestation of a general phenomenon (e.g., related to the definition

of stability) or is it a consequence of a bias in our choice of ontologies for the experiments. On the technical side, further relevance measures are currently under examination, in particular, ones that bring in some semantics either by exploiting the input ontology or by exploring external sources (upper ontologies, structured vocabularies, etc.). We believe that the ultimate filtering method should rely on a combination of such measures, hence at a more advanced stage the exact form of that combination should be investigated.

References

[1] H. Alani, C. Brewster: *Metrics for ranking ontologies*, In: Evaluating Ontologies for the Web Workshop (EON2006), 15th International World Wide Web Conference, 23-26 May 2006, pp. 11-17.

[2] M. Barbut, B. Monjardet: *Ordre et classification: algebre et combinatoire*. Vol- ume 2. Hachette Paris, 1970.

[3] J. Baumeister, D. Seipel: *Verification and refactoring of ontologies with rules*. In: Managing Knowledge in a World of Networks. Springer (2006) pp. 82-95.

[4] M. Dao, M. Huchard, M.R. Hacene, C. Roume, P. Valtchev: *Improving general- ization level in uml models iterative cross generalization in practice*. In: Conceptual Structures at Work. Springer (2004) pp. 346-360.

[5] B. Ganter, R. Wille: Formal Concept Analysis. Mathematical Foundations. Springer, Berlin-Heidelberg-New York, 1999.

[6] R. Godin, P. Valtchev: *Formal concept analysis-based class hierarchy design in object-oriented software development*. In: Formal Concept Analysis. Springer (2005) pp. 304-323.

[7] A. Gomez-Perez: *Ontological engineering: A state of the art*. Expert Update: Knowledge Based Systems and Applied Artificial Intelligence 2(3) (1999) pp. 33-43

[8] T.R. Gruber, et al.: *A translation approach to portable ontology specifications*. Knowledge acquisition 5(2) (1993) pp. 199-220.

[9] S.M. Henshaw, A. El-Masri, P.A. Sage: *Restructuring ontologies through knowl- edge discovery*. In: E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, IEEE (2008) pp. 441-444.

[10] S.O. Kuznetsov: *On stability of a formal concept*. Annals of Mathematics and Artificial Intelligence 49(1-4) (2007) pp. 101-115.

[11] S. Kuznetsov, S. Obiedkov, C. Roth.: *Reducing the representation complexity of lattice-based taxonomies*. In: Conceptual Structures: Knowledge Architectures for Smart Applications. Springer (2007) pp. 241-254.

[12] J. Nanda, T.W. Simpson, S.R. Kumara, S.B. Shooter: *A methodology for prod- uct family ontology development using formal concept analysis and web ontology language*. Journal of computing and information science in engineering 6 (2006) pp. 103-113.

[13] C. Rijsbergen: *Information retrieval*. Butterworths (1975)

[14] C. Roth, S., Obiedkov, D., Kourie, D.: *Towards concise representation for taxonomies of epistemic communities*. In: Concept Lattices and their Applications. Springer (2008) pp. 240-255.

[15] M. Rouane-Hacene, S. Fennouh, R. Nkambou, P. Valtchev: *Refactoring of on- tologies: Improving the design of ontological models with concept analysis*. In: Tools with Arti cial Intelligence (ICTAI). Volume 2., IEEE (2010) pp. 167-172.

[16] M. Rouane-Hacene, P. Valtchev, R. Nkambou: *Supporting ontology design through large-scale fca-based ontology restructuring*. In: Conceptual Structures for Discovering Knowledge. Springer (2011) pp. 257-269.

[17] M. Rouane-Hacene, M. Huchard, A. Napoli, P. Valtchev: *Relational concept analysis: mining concept lattices from multi-relational data*. Annals of Mathematics and Arti cial Intelligence (2013) pp. 1-28

[18] G. Stumme, A. Maedche: *Fca-merge: Bottom-up merging of ontologies*. In: IJCAI. Volume 1. (2001) pp. 225-230.

[19] M.C. Suarez-Figueroa, A. Gomez-Perez: *First attempt towards a standard glossary of ontology engineering terminology*. In the 8th International Conference on Terminology and Knowledge Engineering (2008).

[20] O. Svab-Zamazal, V. Svatek, C. Meilicke, H. Stuckenschmidt: *Testing the impact of pattern-based ontology refactoring on ontology matching results*. In: The 7th International Semantic Web Conference, Citeseer (2008) pp. 240-271.

Department of Computer Science, UQAM, Montreal, Canada
E-mail address: fennouhs@gmail.com
E-mail address: nkambou@gmail.com
E-mail address: rouanehm@gmail.com
E-mail address: valtchev.petko@uqam.ca

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Special Issue 2, 2014
ICFCA 2014: 12th International Conference on Formal Concept Analysis, Cluj-Napoca, June 10-13, 2014

Incremental Computation of Concept Diagrams

FRANCESCO KRIEGEL

Abstract. Suppose a formal context $\mathbb{K} = (G, M, I)$ is given, whose concept lattice $\underline{\mathfrak{B}}(\mathbb{K})$ with an attribute-additive concept diagram is already known, and an attribute column $\mathbb{C} = (G, \{n\}, J)$ shall be inserted to or removed from it. This paper introduces and proves an incremental update algorithm for both tasks.

1. Introduction

Every formal context $\mathbb{K} = (G, M, I)$ can be displayed by means of an (attribute-additive) diagram of its concept lattice $\underline{\mathfrak{B}}(\mathbb{K})$. However, common algorithms focus on the computation of the concept set $\mathfrak{B}(\mathbb{K})$ or the concept neighborhood[1] $\prec$ as a whole, and do not provide any hints how to update the concept set, the concept neighborhood or even the concept diagram[2] upon changes in the underlying formal context.

Thus, each change would require a recomputation of the whole concept diagram. This means that unchanging fragments would be recomputed (which can be expensive), and furthermore it is then even not guaranteed that the unchanged parts of the concept diagram can be recognized as unchanged in the visualization by the user. To overcome this, I investigated the task of inserting or removing an attribute column into or from a formal context while updating the corresponding concept diagram with as little effort or visual changes as

Received by the editors: March 31, 2014.

2010 *Mathematics Subject Classification.* 03G10.

1998 *CR Categories and Descriptors.* I.2.4 Knowledge Representation Formalisms and Methods.

Key words and phrases. Formal Concept Analysis, Concept Diagrams.

[1]The concept set may be ordered by extent inclusion, which yields a complete lattice $\underline{\mathfrak{B}}(\mathbb{K}) = (\mathfrak{B}(\mathbb{K}), \leq)$, see second section or Ganter's book [2] for further details. The concept neighborhood $\prec$ is the reflexive-transitive reduction of the concept order $\leq$.

[2]A concept diagram is a twice labeled directed acyclic graph $(\mathfrak{B}(\mathbb{K}), \prec, \gamma^{-1}, \mu^{-1})$ induced by the neighborhood relation on the concept set, together with a function that maps each node to a position into a vector space, and each node (A, B) is labeled below by all objects $g \in G$, whose object concept $\gamma(g)$ equals (A, B), and dually labeled above by all attributes $m \in M$ with $\mu(m) = (A, B)$.

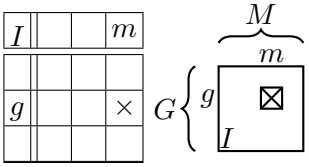
possible. The algorithm is called *iFox*,[3] and could further be used to deduce an update algorithm for setting or deleting just a single incidence entry in $\mathbb{K}$, or for adding or removing a bunch of attribute columns at once, or dualizing it to the insertion or removal of object rows. [4]

The next section gives some preliminaries on basic FCA and some lemmata for context appositions, the third section then formulates the necessary propositions to update the concept set, the neighborhood relation, the labels, the reducibility and seeds (for attributes, when drawing attribute-additive concept diagrams), and the arrow relations, respectivelly. Finally, the algorithm is formulated in pseudo code and its complexity is determined.

All lemmata and theorems can be found in, or are a condensed representation of, [4], except the last proposition describing the incremental update for the down arrows. The references further include some additional hints from the reviewers. This paper does not cover the incremental computation of pseudo-intents or implication bases. If you are interested in this topic, please have a look at Obiedkov and Duquenne's paper [5].

2. Preliminaries

2.1. **Basics of Formal Concept Analysis.** A *formal context* $\mathbb{K} = (G, M, I)$ consists of two sets G (*objects*) and M (*attributes*), and furthermore a binary relation $I \subseteq G \times M$ (*incidence*) between them. For a pair (g, m) that is enclosed in I, we also write gIm and say that object g *has* attribute m (in context $\mathbb{K}$). A common visualization is a *cross table* as shown in the figure below on the left and another one on the right.

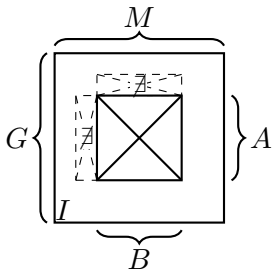


A *formal concept* (A, B) of a context $\mathbb{K}$ consists of two sets, an *extent* $A \subseteq G$ and an *intent* $B \subseteq M$, such that their cartesian product $A \times B$ forms a maximal rectangle within the incidence relation I, more formally

$$A = B^I := \{g \in G \mid \forall_{m \in B}\ gIm\} \text{ and } B = A^I := \{m \in M \mid \forall_{g \in A}\ gIm\}\,.$$

[3]Historical note: In my time at SAP I implemented a FCA library called *fcaFox*, including an *iPred* algorithm. Thus, I chose the name *iFox* for my algorithm for the incremental computation of concept diagrams.

[4]If one wants to dualize the algorithm for row insertion or removal, and the concept diagram is still to be drawn attribute-additivelly, a characterization for the object reducibility update is neccessary. This can be found in [4].



The set of all formal concepts is denoted by $\mathfrak{B}(\mathbb{K})$, and this set can be ordered by means of the extents, i.e. concept (A, B) is smaller than or equals concept (C, D) iff extent A is contained in extent C, symbol: $(A, B) \leq (C, D)$.

$\underline{\mathfrak{B}}(\mathbb{K}) := (\mathfrak{B}(\mathbb{K}), \leq)$ is a complete lattice and its infima and suprema are given by the equations

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

Sometimes the concept lattice of a given formal context shall be visualized for a highly structured and integrated view on its content. For this purpose the definition of a concept lattice is extended to the following notion of a concept diagram.

Definition 1. *Let $\mathbb{K}$ be a formal context and $\underline{V}$ a vector space, e.g. the real plane $\mathbb{R}^2$ or the real space $\mathbb{R}^3$ (or a discrete subset of them like $\mathbb{Z}^2$) for common visualizations. An attribute-additive concept diagram of $\mathbb{K}$ in $\underline{V}$ is a tuple*

$$\underline{\mathfrak{B}}_{\lambda,\sigma}(\mathbb{K}) := (\mathfrak{B}(\mathbb{K}), \prec, \lambda, \sigma)$$

with the following components:

(1) the concept lattice $(\mathfrak{B}(\mathbb{K}), \leq)$ and its neighborhood relation $\prec$,

(2) the default label mapping (other choices possible, e.g. extent cardinality)

$$\lambda\colon \begin{array}{c} \mathfrak{B}(\mathbb{K}) \to \wp(G) \times \wp(M) \\ \mathfrak{b} \mapsto (\{\gamma = \mathfrak{b}\}, \{\mu = \mathfrak{b}\}), \end{array}$$

where all object labels in the first component $\gamma^{-1}(\mathfrak{b})$ are drawn below the concept $\mathfrak{b}$, and dually all attribute labels in the second component $\mu^{-1}(\mathfrak{b})$ are drawn above $\mathfrak{b}$,

(3) and an arbitrary seed vector mapping $\sigma\colon M_{\text{irr}} \to V$.

The position of a concept (A, B) in V is then defined as the sum of the seed vectors of all irreducible attributes in the intent, i.e.

$$\pi(A, B) := \sum_{m \in B \cap M_{\mathrm{irr}}} \sigma(m).$$

2.2. **Appositions of Formal Contexts.** For two formal contexts (G, M, I) and (G, N, J) with disjoint attribute sets $M \cap N = \emptyset$ their *apposition* is defined as

$$(G, M, I)|(G, N, J) := (G, M \dot\cup N, I \dot\cup J).$$

Lemma 2. *Let $(G, M, I)|(G, N, J)$ be an apposition context, then the following equations hold for arbitrary objects $g \in G$ and attributes $m \in M$ and $n \in N$.*

(1) $g(I \dot\cup J)m \Leftrightarrow gIm$ *and*
$g(I \dot\cup J)n \Leftrightarrow gJn$
(2) $g^{I \dot\cup J} = g^I \dot\cup g^J$
(3) $m^{I \dot\cup J} = m^I$ *and*
$n^{I \dot\cup J} = n^J$

Proof. The proof is ommitted here, since the given equations are trivial. □

Lemma 3. *Let $(G, M, I)|(G, N, J)$ be an apposition context and $A \subseteq G$ and $B \subseteq M \dot\cup N$. Then the following equations hold:*

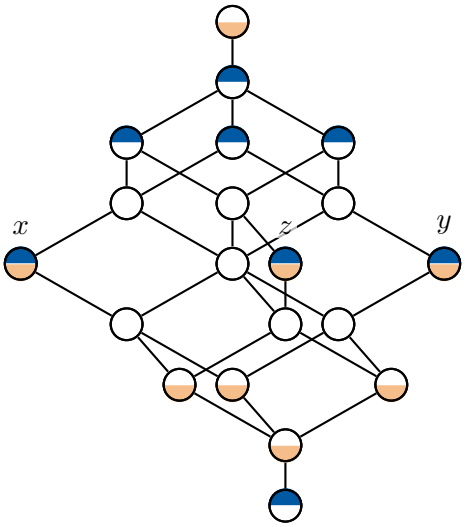
(1) $A^{I \dot\cup J} \cap M = A^I$ *and*
$A^{I \dot\cup J} \cap N = A^J$ *and*
$A^{I \dot\cup J} = A^I \dot\cup A^J$
(2) $(B \cap M)^{I \dot\cup J} = (B \cap M)^I$ *and*
$(B \cap N)^{I \dot\cup J} = (B \cap N)^J$ *and*
$B^{I \dot\cup J} = (B \cap M)^I \cap (B \cap N)^J$
(3) $A^{I(I \dot\cup J)} = (A^{I \dot\cup J} \cap M)^I = A^{II}$ *and*
$A^{J(I \dot\cup J)} = (A^{I \dot\cup J} \cap N)^J = A^{JJ}$ *and*
$A^{(I \dot\cup J)(I \dot\cup J)} = A^{II} \cap A^{JJ}$
(4) $(B \cap M)^{I(I \dot\cup J)} = (B \cap M)^{II} \dot\cup (B \cap M)^{IJ}$ *and*
$(B \cap N)^{J(I \dot\cup J)} = (B \cap N)^{JI} \dot\cup (B \cap N)^{JJ}$ *and*
$B^{(I \dot\cup J)(I \dot\cup J)} = ((B \cap M)^I \cap (B \cap N)^J)^I \dot\cup ((B \cap M)^I \cap (B \cap N)^J)^J$

Proof. The proof is obvious, use 2. □

3. A Very Simple Example

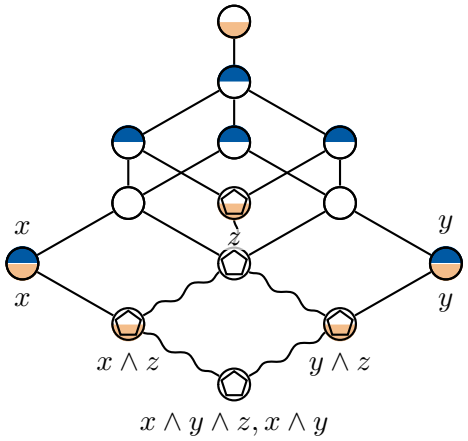
Consider the free distributive lattice FCD(3) with three generating elements x, y, z, as shown in the figure below. An example is constructed that shows how an insertion and a removal of one attribute column affect the concept diagram.

	$x \vee y \vee z$	$x \vee y$	$x \vee z$	$y \vee z$	x	y	z	$\bot$
$x \wedge y \wedge z$	×	×	×	×	×	×	×	↗
$y \wedge z$	×	×	×	×	↗	×	×	
$x \wedge z$	×	×	×	×	×	↗	×	
$x \wedge y$	×	×	×	×	×	×	↗	
z	×	↗	×	×			×	
y	×	×	↗	×		×		
x	×	×	×	↗	×			
$\top$	↗							



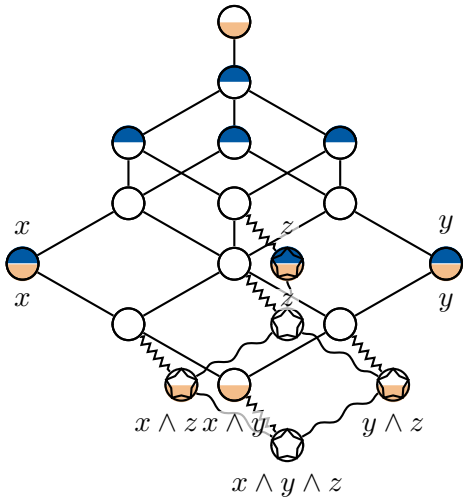
Choose all objects and the first six attributes as old context. The attribute z is to be added. The appropriate contexts and their concept lattices are shown below.

$\mathbb{K}$	$x \vee y \vee z$	$x \vee y$	$x \vee z$	$y \vee z$	x	y
$x \wedge y \wedge z$	×	×	×	×	×	×
$y \wedge z$	×	×	×	×	↗	×
$x \wedge z$	×	×	×	×	×	↗
$x \wedge y$	×	×	×	×	×	×
z	×	↗	×	×		
y	×	×	↗	×		×
x	×	×	×	↗	×	
$\top$	↗					



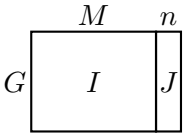
In the initial state above some nodes are marked with a pentagon, these are the generator concepts. The final state below shows the concept lattice after insertion of column z, and the new concept nodes are marked with a star. As you can see the generator structure is locally doubled, and each new concept is a lower neighbor of its generator.

$\mathbb{K}\|\mathbb{C}$	$x \vee y \vee z$	$x \vee y$	$x \vee z$	$y \vee z$	x	y	z
$x \wedge y \wedge z$	×	×	×	×	×	×	×
$y \wedge z$	×	×	×	×	↙	×	×
$x \wedge z$	×	×	×	×	×	↙	×
$x \wedge y$	×	×	×	×	×	×	↙
z	×	↙	×	×			×
y	×	×	↙	×		×	
x	×	×	×	↙	×		
$\top$	↙						



4. Incremental Computation of Concept Diagrams

Throughout the whole section let $\mathbb{K} = (G, M, I)$ be an arbitrary formal context, called *old* context, with its concept diagram $(\mathfrak{B}(\mathbb{K}), \prec, \lambda, \sigma)$. Now the question arises what happens with the concept diagram when a new attribute column is inserted into $\mathbb{K}$, or when an existing attribute column is removed, respectivelly. For this purpose let $n \notin M$ be the *new* attribute with its appropriate *column* context $\mathbb{C} = (G, \{n\}, J)$. The *new context* is then defined as the apposition $\mathbb{K}|\mathbb{C} := (G, M \dot\cup \{n\}, I \dot\cup J)$. [5] [6]



In the ongoing text we analyze the changes that occur on different levels of the concept diagram: concepts, neighborhood, labels, seeds, reducibility and arrows. Most of the main results are displayed in a table style: The old concept diagram on the left side and the new one on the right side, as shown below.

$$(\mathfrak{B}(\mathbb{K}), \prec, \lambda, \sigma) \qquad \leftrightarrows \qquad (\mathfrak{B}(\mathbb{K}|\mathbb{C}), \prec, \lambda, \sigma)$$

[5] For simplification of notion the set parenthesis of the singleton set $\{n\}$ may be omitted: The symbol n is used both for the element n itself and also for a singleton set containing this element n. It is always clear which variant is meant. We thus write $(G, n, J) := (G, \{n\}, J)$ for the column context, and $B \dot\cup n := B \dot\cup \{n\}$ or else $B \setminus n := B \setminus \{n\}$ for an attribute set $B \subseteq M$.

[6] Sometimes both the old context $\mathbb{K}$ and the new context $\mathbb{K}|\mathbb{C}$ share the same set of concept extents; then $\mathbb{C}$ is called *redundant* für $\mathbb{K}$, and *irredundant* otherwise.

Lemma 4. *(1) For all object sets $A \subseteq G$ the following equivalence holds:*

$$A \subseteq n^J \Leftrightarrow A^J = \{n\}\,.$$

(2) For every concept (A, B) of $\mathbb{K}|\mathbb{C}$ it holds that

$$A \subseteq n^J \Leftrightarrow n \in B.$$

Proof.

(1) Let $A \subseteq G$. Trivially $A^J \subseteq \{n\}$ always holds. The other set inclusion follows from the galois property, as $A \subseteq n^J$ is equivalent to $A^J \supseteq \{n\}$.

(2) Let (A, B) be an arbitrary concept of $\mathbb{K}$, i.e. $B = A^{I\dot\cup J} = A^I \mathbin{\dot\cup} A^J$. Then by the first part, $A \subseteq n^J$ holds, iff $A^J = \{n\}$ holds. Obviously this implies $n \in B$. As $n \notin A^I$ always holds, $n \in B$ of course implies $A^J = \{n\}$. □

4.1. **Updating the Formal Concepts.** First, we define a partition of the formal concept set of the old context $\mathbb{K}$, and dually a partition of the formal concept set of the new context $\mathbb{K}|\mathbb{C}$ and then formulate appropriate *update functions*, that map the parts of those partitions to each other. This then fully describes the update mechanism on the concept level from $\mathbb{K}$ to $\mathbb{K}|\mathbb{C}$ and vice versa.

Definition 5. *A concept (A, B) of $\mathbb{K}$ is called*

(1) old concept w.r.t. $\mathbb{C}$, iff its extent is no subset of the new attribute extent, i.e. $A \not\subseteq n^J$,
(2) varying concept w.r.t. $\mathbb{C}$, iff $A \subseteq n^J$, and
(3) generating concept w.r.t. $\mathbb{C}$, iff it is old and $(A \cap n^J)^I = B$ holds.

The set of all old, varying and generating concepts is denoted by $\mathfrak{O}_{\mathbb{C}}(\mathbb{K})$, $\mathfrak{V}_{\mathbb{C}}(\mathbb{K})$ and $\mathfrak{G}_{\mathbb{C}}(\mathbb{K})$. Obviously every $\mathbb{K}$-concept is either old or varying, and each generating concept is particularly an old concept, i.e. $\{\mathfrak{O}_{\mathbb{C}}(\mathbb{K}), \mathfrak{V}_{\mathbb{C}}(\mathbb{K})\}$ is a partition of $\mathfrak{B}(\mathbb{K})$ and $\mathfrak{G}_{\mathbb{C}}(\mathbb{K}) \subseteq \mathfrak{O}_{\mathbb{C}}(\mathbb{K})$ holds.

Definition 6. *A concept (A, B) of $\mathbb{K}|\mathbb{C}$ is called*

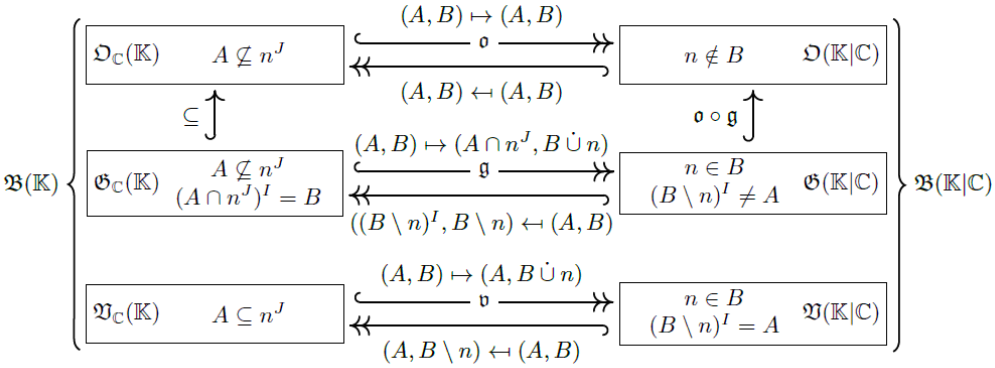
(1) old concept w.r.t. $\mathbb{C}$, iff its intent does not contain the new attribute, i.e. $n \notin B$,
(2) varied concept w.r.t. $\mathbb{C}$, iff $n \in B$ and $(B \setminus n)^I = A$, and
(3) generated (or new) concept w.r.t. $\mathbb{C}$, iff $n \in B$ and $(B \setminus n)^I \neq A$.

The set of old, varied and generated concepts of $\mathbb{K}|\mathbb{C}$ is denoted by $\mathfrak{O}(\mathbb{K}|\mathbb{C})$, $\mathfrak{V}(\mathbb{K}|\mathbb{C})$ and $\mathfrak{G}(\mathbb{K}|\mathbb{C})$. We can easily see, that $\{\mathfrak{O}(\mathbb{K}|\mathbb{C}), \mathfrak{V}(\mathbb{K}|\mathbb{C}), \mathfrak{G}(\mathbb{K}|\mathbb{C})\}$ forms a partition of $\mathfrak{B}(\mathbb{K}|\mathbb{C})$.

As the names suggest, old concepts of $\mathbb{K}$ determine old concepts of $\mathbb{K}|\mathbb{C}$ and vice versa, $\mathbb{K}$-varying concepts determine $\mathbb{K}|\mathbb{C}$-varied concepts, and generating

concepts from $\mathbb{K}$ induce new concepts of $\mathbb{K}|\mathbb{C}$. This is due to the following three bijections.

Lemma 7. *The following three mappings* $\mathfrak{o}$, $\mathfrak{g}$ *and* $\mathfrak{v}$ *are bijections.*



Proof. Each of the following parts prove, that the mentioned mappings are well-defined and bijective. The original proof in [4] used the nested concept lattice of $\mathbb{C}$ in $\mathbb{K}$, the presented proof here is much simpler.

(1) The mapping $\mathfrak{o}$ and its inverse are well-defined by lemma 4. The lower mapping is indeed the inverse, as we can easily see.

(2) Let (A,B) be a generating concept of $\mathbb{K}$ w.r.t. $\mathbb{C}$, then

$$(A \cap n^J)^{I \dot\cup J} = (A \cap n^J)^I \dot\cup (A \cap n^J)^J = B \dot\cup \{n\}$$

as surely $n \in (A \cap n^J)^J$ holds (because every object in $A \cap n^J$ has the new attribute n w.r.t. J), and

$$(B \dot\cup \{n\})^{I \dot\cup J} = B^I \cap n^J = A \cap n^J.$$

Thus, the mapping $\mathfrak{g}$ is well-defined. The lower mapping is also well-defined by the following observation for an arbitrary generated concept (A,B) of $\mathbb{K}|\mathbb{C}$, see also lemma 3

$$(B \setminus \{n\})^{II} = (B \cap M)^{II} = (A^{I \dot\cup J} \cap M)^{II} = A^{III} = A^I = \cdots = B \setminus \{n\}$$

Both mappings are inverse to each other, as can be seen on the intents.

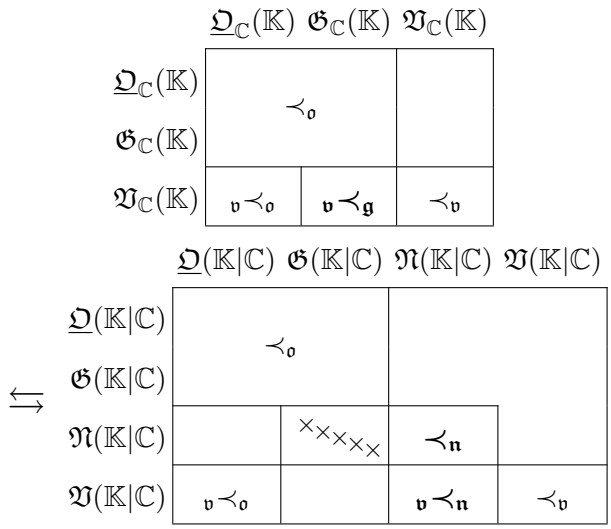
(3) Let (A,B) be a varying concept of $\mathbb{K}$ w.r.t. $\mathbb{C}$, then for the extent we have $A^{I \dot\cup J} = A^I \dot\cup A^J = B \dot\cup \{n\}$ and for the intent we infer $(B \dot\cup \{n\})^{I \dot\cup J} = B^I \cap n^J = A \cap n^J = A$. Conversely for the lower mapping it holds that $A^I = A^{I \dot\cup J} \cap M = B \cap M = B \setminus \{n\}$ and $(B \setminus \{n\})^I = A$ by assumption. Both mappings are mutually inverse by looking on the extents. □

4.2. **Updating the Neighborhood.** Of course, when visualizing concept lattices, it is neccessary to update the concept neighborhood relation as well. Some first investigations show that there are blocks within the neighborhood that do not change from $\mathbb{K}$ to $\mathbb{K}|\mathbb{C}$ and vice versa. [7]

When inserting the new attribute, mainly the lower neighbors of the new concepts have to be computed. It is already clear that each new concept must be a lower neighbor of its generating concept. Also, each varied concept can not have any generator concept as upper neighbor.

For the attribute removal the columns and rows of new concepts of $\mathbb{K}|\mathbb{C}$ are just deleted, and the neighborhood between the varying and generator concepts needs to be determined.

A complete overview is given in the following figure (the bold subrelations change, and have to be computed; all other parts may be copied).



Within the figure the *really old* concepts are used, that are just the old concepts which are no generator concept, denoted by

$$\underline{\mathfrak{O}}(\mathbb{K}|\mathbb{C}) := \mathfrak{O}(\mathbb{K}|\mathbb{C}) \setminus \mathfrak{G}(\mathbb{K}|\mathbb{C}) \qquad \text{and} \qquad \underline{\mathfrak{O}}_{\mathbb{C}}(\mathbb{K}) := \mathfrak{O}_{\mathbb{C}}(\mathbb{K}) \setminus \mathfrak{G}_{\mathbb{C}}(\mathbb{K}).$$

Theorem 8. *The concept neighborhood relation only changes partially:*

(1) Let $\mathfrak{a}, \mathfrak{b}$ *be two generators in* $\mathbb{K}$ *w.r.t.* $\mathbb{C}$*, then* $\mathfrak{n}(\mathfrak{a}) \prec_{\mathfrak{n}} \mathfrak{n}(\mathfrak{b})$ *holds, iff*

$$[\mathfrak{a}, \mathfrak{b}] \cap \mathfrak{G}_{\mathbb{C}}(\mathbb{K}) = \{\mathfrak{a}, \mathfrak{b}\},$$

i.e. when there is no generating concept between $\mathfrak{a}$ *and* $\mathfrak{b}$*.*

[7] It easy to see that the neighborhood between old concepts does not change, and so also for the varying/varied concepts.

(2) *If* $\mathfrak{a}$ *is varying and* $\mathfrak{b}$ *a generator, both in* $\mathbb{K}$ *w.r.t.* $\mathbb{C}$*, then* $\mathfrak{v}(\mathfrak{a}) \prec_{\mathfrak{v}\mathfrak{n}} \mathfrak{n}(\mathfrak{b})$ *holds iff*

$$[\mathfrak{a}, \mathfrak{b}] \cap \mathfrak{G}_{\mathbb{C}}(\mathbb{K}) \cap \mathfrak{V}_{\mathbb{C}}(\mathbb{K}) = \{\mathfrak{a}, \mathfrak{b}\},$$

so if there is no generator or varying concept between $\mathfrak{a}$ *and* $\mathfrak{b}$*.*

(3) *Let* $\mathfrak{a}$ *be a varied concept and* $\mathfrak{b}$ *a new concept in* $\mathbb{K}|\mathbb{C}$*. Then* $\mathfrak{v}^{-1}(\mathfrak{a}) \prec_{\mathfrak{v}\mathfrak{g}} \mathfrak{g}(\mathfrak{b})$ *holds in* $\underline{\mathfrak{B}}(\mathbb{K})$ *iff* $\mathfrak{a} \prec_{\mathfrak{v}\mathfrak{n}} \mathfrak{b}$ *and*

$$(\mathfrak{a}, \mathfrak{o}\mathfrak{g}(\mathfrak{b})) \cap \underline{\mathfrak{O}}(\mathbb{K}|\mathbb{C}) = \emptyset.$$

Proof. It is simply a proof by cases. The proof for the unchanging components is ommited here, and only the changing fragments are investigated. Some first clues can be obtained from the neighborhood structure within the nested concept lattice.

(1) Let first $\mathfrak{a}$ and $\mathfrak{b}$ be two generating concepts. When are their generated new concepts neighboring? This can only be the case when no other concept is between them, and the only type of concept fitting between two new concepts is another new concept. In summary, the corresponding new concepts $\mathfrak{n}(\mathfrak{a})$ and $\mathfrak{n}(\mathfrak{b})$ are neighbors, iff there is no other generator concept between $\mathfrak{a}$ and $\mathfrak{b}$.

(2) Analogously, let $\mathfrak{a}$ be a varying concept and $\mathfrak{b}$ a generating concept. Then the varied concept $\mathfrak{v}(\mathfrak{a})$ can only be covered by the new concept $\mathfrak{g}(\mathfrak{b})$, when there is no other $\mathbb{K}|\mathbb{C}$-concept between them. There could only be a varied or a new concept between them, and thus the statement holds exactly when there is no generator or varying concept between $\mathfrak{a}$ and $\mathfrak{b}$.

(3) This is an immediate consequence of 2. For a varied concept $\mathfrak{a}$ and a new concept $\mathfrak{b}$, the corresponding varying concept $\mathfrak{v}^{-1}(\mathfrak{a})$ can only be covered by the generating concept $\mathfrak{g}(\mathfrak{b})$, when there is (in addition to the condition from 2) no really old concept between $\mathfrak{v}^{-1}(\mathfrak{a})$ and $\mathfrak{g}(\mathfrak{b})$, since this is the only missing concept type in the characterization of neighboring varied and new concepts, see 2. □

4.3. **Updating the Labels.** Each concept node is labeled with some objects and attributes. More exactly, each object concept (g^{II}, g^I) where $g \in G$ is labeled with g above, and dually every attribute concept (m^I, m^{II}) where $m \in M$ is labeled with m below.

When changing the context by column insertion or removal, the attribute label n must be inserted in or removed from the concept diagram, and furthermore some other already existing labels might have to be moved to other concept nodes. In detail, the object concepts $\gamma(g)$ and the attribute concepts $\mu(m)$ have to be investigated to characterize the label update for the column

insertion or removal. A complete overview for this is given in [4], and the condensed result is presented in the following proposition.

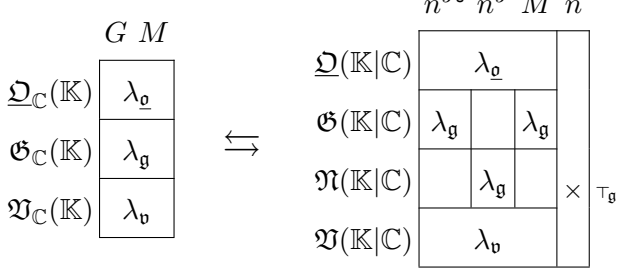
Proposition 9. *(1) When adding the new attribute n, there must be an corresponding attribute concept $\mu(n)$ that is labeled with n. If n is not redundant, then this new concept is always generated by the greatest generator concept*

$$\top_{\mathfrak{g}} := \bigvee \mathfrak{G}_{\mathbb{C}}(\mathbb{K}) = (n^{JII}, n^{JI}),$$

and then $\mu(n) = \mathfrak{n}(\top_{\mathfrak{g}})$ holds.

(2) For the concept diagram transition from $\mathbb{K}$ to $\mathbb{K}|\mathbb{C}$ only object labels at previously generator nodes can move downwards to the corresponding new concept node. No attribute labels change.

(3) Vice versa, for the transition from $\mathbb{K}|\mathbb{C}$ back to $\mathbb{K}$ the attribute label n is removed and the object labels of a generator concept are merged with the object labels of the approriate new concept, i.e. let (A, B) be a generator with object labels L and (C, D) the generated new concept with object labels M, then (A, B) is labeled with each element from the union $L \cup M$ in the old concept diagram.



Proof. This is easy and straight-forward by analyzing the object and attribute concepts, and determining whether they are old, varying/varied or generating/new. □

4.4. **Updating the Reducibility and Seeds.** In order to maximize the quality of an attribute-additive concept diagram it is important to know the irreducible attributes of the context. Each attribute can then be displayed as the infimum of irreducible attributes, and thus, the set of irreducible attributes spans the whole concept diagram and it suffices to assign seed vectors just to the irreducible attributes. Of course, when inserting or removing $\mathbb{C}$ to $\mathbb{K}$ or from $\mathbb{K}|\mathbb{C}$, the attribute irreducibility may change for the existing attributes.

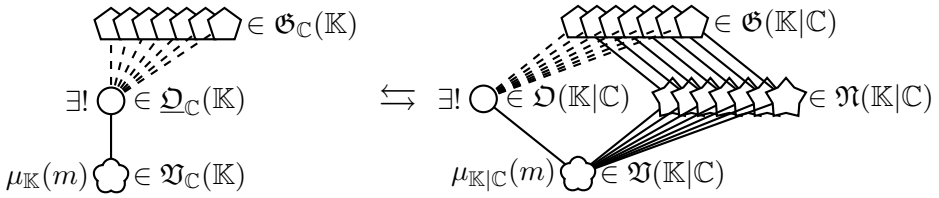
Proposition 10. *The attribute reducibility can be updated via the following observations:*

(1) Each $\mathbb{K}$-reducible attribute is also $\mathbb{K}|\mathbb{C}$-reducible.

(2) A $\mathbb{K}$-irreducible attribute $m \in M$ is $\mathbb{K}|\mathbb{C}$-reducible, iff its $\mathbb{K}$-attribute concept is varying and the corresponding unique upper neighbor $\mu^*_{\mathbb{K}}(m)$ is really old, and furthermore at least one superconcept of $\mu^*_{\mathbb{K}}(m)$ is a generator concept.

(3) Every $\mathbb{K}|\mathbb{C}$-irreducible attribute is also $\mathbb{K}$-irreducible.

(4) A $\mathbb{K}|\mathbb{C}$-reducible attribute $m \in M$ is $\mathbb{K}$-irreducible, iff its $\mathbb{K}|\mathbb{C}$-attribute concept is varied and has exactly one old upper neighbor $\mathfrak{b}$ and overthis only new upper neighbors, that are generated from superconcepts of $\mathfrak{b}$.



Proof.

(1) First, if m is a $\mathbb{K}$-reducible attribute, then the attribute extent m^I can be obtained by an intersection of attribute extents $\bigcap_{m\in B} m^I$ with $m \notin B$. Obviously then also

$$m^{(I \dot\cup J)} = m^I = \bigcap_{m\in B} m^I = \bigcap_{m\in B} m^{(I\dot\cup J)}$$

holds, hence m is $\mathbb{K}|\mathbb{C}$-reducible.

(2) Second, let m be a $\mathbb{K}$-irreducible attribute.

($\Rightarrow$) Suppose m is $\mathbb{K}|\mathbb{C}$-reducible. If $\mu_{\mathbb{K}}(m)$ were an old concept, then $\mu_{\mathbb{K}|\mathbb{C}}(m) = \mathfrak{o}(\mu_{\mathbb{K}}(m))$ and the set of upper neighbors does not change according to theorem 8. Thus, the irreducibility of m in $\mathbb{K}$ implies the irreducibility of m in $\mathbb{K}|\mathbb{C}$. Contradiction! Hence, the attribute concept $\mu_{\mathbb{K}}(m)$ must be varying. By 7, there are no other old or varied upper neighbors of $\mu_{\mathbb{K}|\mathbb{C}}(m)$. If $\mu^*_{\mathbb{K}}(m)$ would be a varying or generating concept, then

$$\mu_{\mathbb{K}|\mathbb{C}}(m) = \mathfrak{v}(\mu_{\mathbb{K}}(m)) \prec \begin{cases} \mathfrak{v}(\mu^*_{\mathbb{K}}(m)) & \text{if } \mu^*_{\mathbb{K}}(m) \in \mathfrak{V}_{\mathbb{C}}(\mathbb{K}) \\ \mathfrak{g}(\mu^*_{\mathbb{K}}(m)) & \text{if } \mu^*_{\mathbb{K}}(m) \in \mathfrak{G}_{\mathbb{C}}(\mathbb{K}) \end{cases}$$

holds. Let $\mathfrak{b} \in \mathfrak{G}_{\mathbb{C}}(\mathbb{K})$ with $\mathfrak{b} \neq \mu^*_{\mathbb{K}}(m)$, such that $\mathfrak{g}(\mathfrak{b})$ covers $\mu_{\mathbb{K}|\mathbb{C}}(m)$, then $\mu_{\mathbb{K}}(m)$ must be a lower neighbor of $\mathfrak{b}$ and there is no varying or generating concept between them. So $\mu_{\mathbb{K}}(m) \prec \mu^*_{\mathbb{K}}(m) < \mathfrak{b}$ must hold, but this is a contradiction. In summary, $\mathfrak{v}(\mu^*_{\mathbb{K}}(m))$ or $\mathfrak{g}(\mu^*_{\mathbb{K}}(m))$, respectivelly, must be the unique upper neighbor of $\mu_{\mathbb{K}|\mathbb{C}}(m)$, and m would be $\mathbb{K}|\mathbb{C}$-irreducible. Contradiction! Hence $\mu^*_{\mathbb{K}}(m)$ must be an old non-generator concept.

Finally if there were no generating superconcept above $\mu^*_{\mathbb{K}}(m)$, then $\mathfrak{o}(\mu^*_{\mathbb{K}}(m))$ were the only upper neighbor of $\mu_{\mathbb{K}|\mathbb{C}}(m)$, *i.e.* m would be $\mathbb{K}|\mathbb{C}$-irreducible. Contradiction!

($\Leftarrow$) Suppose the attribute concept $\mu_{\mathbb{K}}(m)$ is a varying concept and its unique upper neighbor $\mu^*_{\mathbb{K}}(m)$ is an old non-generator concept that has at least one generator superconcept. Denote the minimal ones of these generator superconcepts by $\xi_1, \xi_2, \ldots, \xi_k$. Then the following structure on the left side can be found within the concept lattice of $\mathbb{K}$. Neighboring concept nodes are connected by straight line segments and comparable concepts are connected by zig zag line segments. Then according to theorem 8 the new concepts $\mathfrak{g}(\xi_1), \ldots, \mathfrak{g}(\xi_k)$ must cover the varied attribute concept $\mathfrak{v}(\mu_{\mathbb{K}}(m))$. This is due to the fact, that no varying concept can be greater than an old concept, and the generators $\xi_1, \ldots, \xi_k$ are minimal. Furthermore $\mu^*_{\mathbb{K}}(m)$ is the unique upper neighbor of $\mu_{\mathbb{K}}(m)$, hence there cannot be any varying or generating concept between $\mu_{\mathbb{K}}(m)$ and each ξ_j. In summary, the transition from $\mathbb{K}$ to $\mathbb{K}|\mathbb{C}$ changes the concept lattice structure as displayed in the right diagram. Obviously $\mu_{\mathbb{K}|\mathbb{C}}(m) = \mathfrak{v}(\mu_{\mathbb{K}}(m))$ has more than one upper neighbor, hence m is $\mathbb{K}|\mathbb{C}$-reducible.

(3) Let first $m \in M$ be a $\mathbb{K}|\mathbb{C}$-irreducible attribute. Then m must also be $\mathbb{K}$-irreducible, as otherwise m were $\mathbb{K}|\mathbb{C}$-irreducible by 1.

(4) Second, let $m \in M$ be $\mathbb{K}|\mathbb{C}$-reducible attribute.

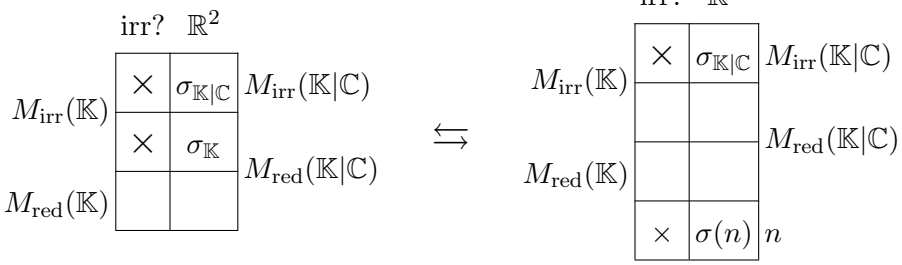
($\Rightarrow$) Suppose m is $\mathbb{K}$-irreducible. Then $\mu_{\mathbb{K}|\mathbb{C}}(m)$ must be a varied concept. Otherwise $\mu_{\mathbb{K}}(m) = \mathfrak{o}^{-1}(\mu_{\mathbb{K}|\mathbb{C}}(m))$ were an old concept and this is a contradiction to 1. If $\mu_{\mathbb{K}|\mathbb{C}}(m)$ had more than one old (and thus non-generating) upper neighbor in $\mathfrak{B}(\mathbb{K}|\mathbb{C})$, then the according old concepts in $\mathfrak{B}(\mathbb{K})$ would cover $\mu_{\mathbb{K}}(m)$. This is a contradiction to the $\mathbb{K}$-irreducibility of m. So $\mu_{\mathbb{K}|\mathbb{C}}(m)$ has exactly one old upper neighbor $\omega \in \mathfrak{O}(\mathbb{K}|\mathbb{C})$, all other upper neighbors must be varied or new concepts. If a varied concept covers $\mu_{\mathbb{K}|\mathbb{C}}(m)$, then its appropriate varying concept covers $\mu_{\mathbb{K}}(m)$ as well. Again, this is a contradiction to the $\mathbb{K}$-irreducibility. So all other upper neighbors must be new concepts. If there were any new concept $\nu \in \mathfrak{G}(\mathbb{K}|\mathbb{C})$ whose generator ξ is not a superconcept of ω, then $\mu_{\mathbb{K}}(m)$ would be covered by $\mathfrak{o}^{-1}(\xi)$. Then $\mu_{\mathbb{K}}(m)$ had at least two upper neighbors and this contradicts the $\mathbb{K}$-irreducibility.

($\Leftarrow$) Suppose $\mu_{\mathbb{K}|\mathbb{C}}(m)$ varies and has exactly one upper neighbor ω and overthis only new upper neighbors $\nu_1, \ldots, \nu_k$, whose generators are greater than ω. Then choose $\xi_j := \mathfrak{g}(\nu_j)$ and the same structure

as in the right diagram above occurs, and by theorem 8 $\mathfrak{o}^{-1}(\omega) = \mu^*_{\mathbb{K}}(m)$ must be the unique upper neighbor of $\mu_{\mathbb{K}}(m)$. This means m is $\mathbb{K}$-irreducible. □

The update of the seed map can now be done with the following rules.

(1) When adding the new column, delete the seeds for $\mathbb{K}|\mathbb{C}$-reducible attributes, that were $\mathbb{K}$-irreducible, and introduce a new seed for n.
(2) When removing the column, delete the seed for n and compute seeds for the previously reducible attributes in $\mathbb{K}|\mathbb{C}$, which are now irreducible in $\mathbb{K}$.

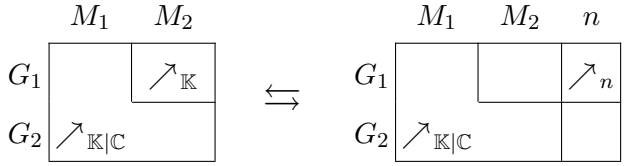


5. Incremental Computation of the Arrow Relations

5.1. **Updating the Up Arrows.** This section investigates the changes for the up arrow relation. For this purpose the object set and the attribute set is splitted into the following components:

$$G_1 := \left\{g \;\middle|\; g \notin n^J\right\},\ G_2 := \left\{g \;\middle|\; g \in n^J\right\},\ \text{and}$$
$$M_1 := \left\{m \;\middle|\; m^I \not\subset n^J\right\},\ M_2 := \left\{m \;\middle|\; m^I \subset n^J\right\}$$

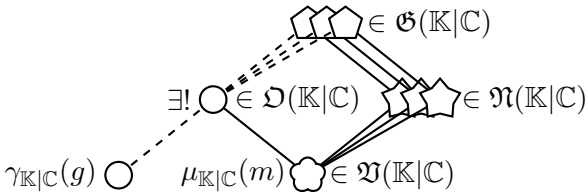
When the column is inserted the block $\nearrow_{\mathbb{K}} \subseteq G_1 \times M_2$ can simply be deleted. The only entries to compute is the upper column $\nearrow_n \subseteq G_1 \times \{n\}$. [8] It is even possible to give a characterization for the $\nearrow_{\mathbb{K}}$ block for the column removal.



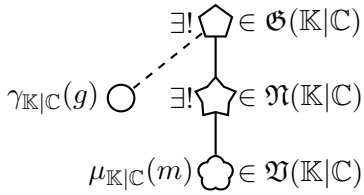
Proposition 11. *(1) Up arrows in $\mathbb{K}$ and $\mathbb{K}|\mathbb{C}$ may only differ on the subset $G_1 \times M_2$ and $G_1 \times \{n\}$. All other parts are equal.*
(2) Let $g \in G_1$ and $m \in M_2$, then $g \nearrow_{\mathbb{K}} m$ holds, iff one of the following conditions is fulfilled:

[8] Of course, there cannot be any arrows in the lower column $G_2 \times \{n\}$ as it is full of crosses.

(a) m is $\mathbb{K}|\mathbb{C}$-reducible, and its attribute concept $\mu_{\mathbb{K}|\mathbb{C}}(m) \in \mathfrak{V}(\mathbb{K}|\mathbb{C})$ has exactly one old upper neighbor $\mathfrak{b}$ and overthis only new upper neighbors generated by superconcepts of $\mathfrak{b}$, and furthermore $\gamma_{\mathbb{K}|\mathbb{C}}(g)$ is a subconcept of $\mathfrak{b}$.



(b) m is $\mathbb{K}|\mathbb{C}$-irreducible, $\mu^*_{\mathbb{K}|\mathbb{C}}(m) \in \mathfrak{N}(\mathbb{K}|\mathbb{C})$ and the old object concept $\gamma_{\mathbb{K}|\mathbb{C}}(g) \in \mathfrak{O}(\mathbb{K}|\mathbb{C})$ is a subconcept of the generator $\mathfrak{og}(\mu^*_{\mathbb{K}|\mathbb{C}}(m))$.



Proof.

(1) This is obvious.

(2) In case $g \in n^J$ this follows from the preceding lemma as well. Suppose $g \notin n^J$. Then the object concept of g in $\mathbb{K}|\mathbb{C}$ is given by

$$\gamma_{\mathbb{K}|\mathbb{C}}(g) = \begin{cases} \mathfrak{o}(\gamma_{\mathbb{K}}(g)) & \text{if } \gamma_{\mathbb{K}}(g) \in \mathfrak{O}_{\mathbb{C}}(\mathbb{K}) \\ \mathfrak{v}(\gamma_{\mathbb{K}}(g)) & \text{if } \gamma_{\mathbb{K}}(g) \in \mathfrak{V}_{\mathbb{C}}(\mathbb{K}) \end{cases}.$$

(a) Let m be $\mathbb{K}|\mathbb{C}$-reducible. $g \nearrow_{\mathbb{K}} m$ can only hold, when m is irreducible in $\mathbb{K}$, *i.e.* when $\mu_{\mathbb{K}|\mathbb{C}}(m) \in \mathfrak{V}(\mathbb{K}|\mathbb{C})$ has exactly one old upper neighbor ω and overthis only new upper neighbors, whose generators are superconcepts of ω, according to 10. Then $\mathfrak{o}^{-1}(\omega)$ is the unique upper neighbor of $\mu_{\mathbb{K}}(m)$. Furthermore, $\gamma_{\mathbb{K}|\mathbb{C}}(g) \leq \omega$ holds, iff $\gamma_{\mathbb{K}}(g) \leq \mu^*_{\mathbb{K}}(m)$, *i.e.* iff $g \nearrow_{\mathbb{K}} m$.

(b) When m is $\mathbb{K}|\mathbb{C}$-irreducible, then m is also $\mathbb{K}$-irreducible by 10. Furthermore, $g \notin n^J$ implies $g \not\nearrow_{\mathbb{K}|\mathbb{C}} m$, *i.e.* $\gamma_{\mathbb{K}|\mathbb{C}}(g)$ is no subconcept of $\mu^*_{\mathbb{K}|\mathbb{C}}(m)$. If $\mu^*_{\mathbb{K}|\mathbb{C}}(m)$ is an old concept, then $\mathfrak{o}^{-1}(\mu^*_{\mathbb{K}|\mathbb{C}}(m))$ is the unique upper neighbor of

$$\mu_{\mathbb{K}}(m) = \begin{cases} \mathfrak{o}^{-1}(\mu_{\mathbb{K}|\mathbb{C}}(m)) & \text{if } \mu_{\mathbb{K}|\mathbb{C}}(m) \in \mathfrak{O}(\mathbb{K}|\mathbb{C}) \\ \mathfrak{v}^{-1}(\mu_{\mathbb{K}|\mathbb{C}}(m)) & \text{if } \mu_{\mathbb{K}|\mathbb{C}}(m) \in \mathfrak{V}(\mathbb{K}|\mathbb{C}) \end{cases}.$$

Then $\gamma_{\mathbb{K}}(g)$ is a subconcept of $\mu^*_{\mathbb{K}}(m)$, iff $\gamma_{\mathbb{K}|\mathbb{C}}(g)$ is a subconcept of $\mu^*_{\mathbb{K}|\mathbb{C}}(m)$. As this cannot occur according to the preconditions, $g \not\nearrow_{\mathbb{K}} m$ must hold. If $\mu^*_{\mathbb{K}|\mathbb{C}}(m)$ is a varied concept, then $\mathfrak{v}^{-1}(\mu^*_{\mathbb{K}|\mathbb{C}}(m))$ is the unique upper neighbor of $\mu_{\mathbb{K}}(m) = \mathfrak{v}^{-1}(\mu_{\mathbb{K}|\mathbb{C}}(m))$. Then $\gamma_{\mathbb{K}}(g)$ is smaller than $\mu^*_{\mathbb{K}}(m)$, iff $\gamma_{\mathbb{K}|\mathbb{C}}(g)$ is a subconcept of $\mu^*_{\mathbb{K}|\mathbb{C}}(m)$. Thus, $g \not\nearrow_{\mathbb{K}} m$ as well in this case. If the unique upper neighbor $\mu^*_{\mathbb{K}|\mathbb{C}}(m)$ is a new concept, then according to 8 $\mathfrak{g}(\mu^*_{\mathbb{K}|\mathbb{C}}(m))$ must be the unique upper neighbor of $\mu_{\mathbb{K}}(m) = \mathfrak{v}^{-1}(\mu_{\mathbb{K}|\mathbb{C}}(m))$. Furthermore $\gamma_{\mathbb{K}}(g)$ can only be a subconcept of $\mu^*_{\mathbb{K}}(m)$, if it is an old concept and a subconcept of the generator. (If $\gamma_{\mathbb{K}}(g)$ would be varying and smaller than the generator, $\gamma_{\mathbb{K}|\mathbb{C}}(g)$ must be smaller than the new generated concept as well, in contradiction to the preconditions.) In summary, $g \nearrow_{\mathbb{K}} m$ holds in this case, iff $\gamma_{\mathbb{K}|\mathbb{C}}(g)$ is an old concept and smaller than the generator of the upper neighbor of $\mu_{\mathbb{K}|\mathbb{C}}(m)$. □

5.2. **Updating the Down Arrows.** Suppose, $g \in G$ is an object and $m \in M$ is an attribute of $\mathbb{K}$. First, observe that by definition of the down arrows it holds that

$$g \swarrow_{\mathbb{K}} m \Leftrightarrow g \not I\, m \text{ and } \underset{h \in G}{\forall} g^I \subsetneq h^I \Rightarrow h\ I\ m$$

and analogously

$$g \swarrow_{\mathbb{K}|\mathbb{C}} m \Leftrightarrow \underbrace{(g,m) \notin (I \dot\cup J)}_{\Leftrightarrow g \not I m} \text{ and } \underset{h \in G}{\forall} g^{I \dot\cup J} \subsetneq h^{I \dot\cup J} \Rightarrow \underbrace{h\ (I \dot\cup J)\ m}_{hIm}.$$

Proposition 12. *(1) When $g \swarrow_{\mathbb{K}|\mathbb{C}} m$ holds, then also $g \swarrow_{\mathbb{K}} m$ holds.*

(2) Let $g \swarrow_{\mathbb{K}} m$ where $g \not I\, n$. Then $g \swarrow_{\mathbb{K}|\mathbb{C}} m$ holds, if there is no $\mathbb{K}$-equivalent object h (i.e. $g^I = h^I$), which is not $\mathbb{K}|\mathbb{C}$-equivalent to g (i.e. $h\ J\ n$).

(3) Let $g \swarrow_{\mathbb{K}} m$ where $g\ J\ n$. Then $g \swarrow_{\mathbb{K}|\mathbb{C}} m$ holds, if each object $h \in G$ with $g^I \subsetneq h^I$ also has the new attribute n.

Proof.

(1) This is obvious, since $g^I \subsetneq h^I$ implies $g^{(I \dot\cup J)} \subsetneq h^{(I \dot\cup J)}$.

(2) Suppose g does not have the new attribute n, and $g \swarrow_{\mathbb{K}} m$ holds. When does $g \swarrow_{\mathbb{K}|\mathbb{C}} m$ also hold? For $h \in G$ with $g^{I \dot\cup J} \subsetneq h^{I \dot\cup J}$ it holds that $g^I \subsetneq h^I \dot\cup h^J$.
   - If $g^I \subsetneq h^I$, then $h\ I\ m$ holds since $g \swarrow_{\mathbb{K}} m$.
   - If $g^I = h^I$ and $h\ J\ n$, then $h \not I\, m$ since g does not have m (as $g \swarrow_{\mathbb{K}} m$ holds).

Obviously $g \swarrow_{\mathbb{K}|\mathbb{C}} m$ cannot hold, when the second condition is fulfilled.

(3) Finally, let g have the new attribute n and $g \swarrow_{\mathbb{K}} m$. To check, whether $g \swarrow_{\mathbb{K}|\mathbb{C}} m$ hold, let $h \in G$ be an object, whose $\mathbb{K}|\mathbb{C}$-intent is a proper superset of $g^{I \dot{\cup} J}$. It then easily follows, that also h must have the new attribute n and $g^I \subsetneq h^I$ must hold for the $\mathbb{K}$-intents. By the precondition this yields $h\ I\ m$. Since this is true for all such objects h, $g \swarrow_{\mathbb{K}|\mathbb{C}} m$ can be concluded. □

6. Conclusion

This document described an update algorithm for the insertion or removal of an attribute column to or from a formal context, whose concept diagram is already known. It has been implemented in *ConceptExplorer FX*, that is a partial re-implementation of the well-known FCA tool *ConceptExplorer* by Serhiy Yevtushenko et al.

The introduced lemmata and propositions may be extended for the insertion or removal of several attribute columns at once, or it may be dualized for object row insertion or deletion, as also suggested in the introduction. Furthermore it may be possible to generalize it to insert elements into an arbitrary complete lattice, not only to insert new attribute concepts into a concept lattice (and also for deletion, of course).

References

[1] C. Carpineto, G. Romano, *Concept Data Analysis : Theory and Applications*. Wiley, 2004.

[2] B. Ganter, R. Wille, *Formal Concept Analysis: Mathematical Foundations*. Springer, 1998.

[3] R. Missaoui, R. Godin, H. Alaoui, *Incremental concept formation algorithms based on galois (concept) lattices*. in Computational Intelligence, 11 (1995), pp.246-267.

[4] F. Kriegel. *Visualization of conceptual data with methods of formal concept analysis*. Master's thesis, Technische Universitat Dresden, Faculty of Mathematics, Institute for Algebra, SAP AG, Research Center Dresden, 2012.

[5] S A. Obiedkov, V. Duquenne, *Attribute-incremental construction of the canonical implication basis*. Ann. Math. Artif. Intell., 49 (2007), pp. 77-99.

[6] M. Skorsky. *Endliche Verbande - Diagramme und Eigenschaften*. PhD thesis, 1992.

[7] P. Valtchev, R. Missaoui, P. Lebrun, *A partition-based approach towards constructing galois (concept) lattices*, Discrete Mathematics, 256(2002), pp.801-829.

Theoretical Computer Science, TU Dresden, Germany
E-mail address: francesco.kriegel@tu-dresden.de

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Special Issue 2, 2014
ICFCA 2014: 12th International Conference on Formal Concept Analysis, Cluj-Napoca, June 10-13, 2014

RANKING FORMAL CONCEPTS BY UTILIZING MATRIX FACTORIZATION

LENKA PISKOVÁ, TOMÁŠ HORVÁTH, AND STANISLAV KRAJČI

Abstract. Formal Concept Analysis often produce huge number of formal concepts even for small input data. Such a large amount of formal concepts, which is intractable to analyze for humans, calls for a kind of a ranking of formal concepts according to their importance in the given application domain. In this paper, we propose a novel approach to rank formal concepts that utilizes matrix factorization, namely, a mapping of objects and attributes to a common latent space. The lower the distance between objects and/or attributes in the extent and/or intent of a formal concept in the latent space of factors, the more important the formal concept is considered to be. We provide an illustrative example of our approach and examine the impact of various matrix factorization techniques using real-world benchmark data.

1. Introduction

Formal Concept Analysis (FCA) [9] is a method for analyzing object-attribute data. In the basic setting, table entries are 1 or 0 indicating whether an object has a given attribute or not. FCA aims at finding so-called formal concepts (as well as the subconcept-superconcept relation among them) from this data. A formal concept is a formalization of the concept of a 'concept' which consists of two parts, a set of objects which forms its extension and a set of attributes which forms its intension [16]. The set of all concepts ordered by $\leq$ forms a complete lattice [9].

One of the obstacles in real-world application of FCA is that it often produces a huge number of formal concepts which can be exponential in the size of input data (see Table 3).

A kind of a ranking of resulting formal concepts would be beneficial for a human expert in the process of analyzing the formal concepts. We think

Received by the editors: 25 March 2014.

2010 *Mathematics Subject Classification.* 06-XX, 06Bxx.

1998 *CR Categories and Descriptors.* I.2.m [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE – *Miscellaneous* .

Key words and phrases. Formal Concept Analysis, formal concept, coherence, matrix factorization.

that such a ranking is domain and/or task specific and strongly depends on the actual needs of a user (i.e. a domain expert which is intended to use the outputs of FCA). Because of this, an approach to rank formal concepts should be intuitive, easily understandable and provide sufficient insight for a user.

In this paper, we propose a novel approach to rank formal concepts that utilizes matrix factorization, namely, a mapping of objects and attributes to a common latent space. The lower the distance between objects and/or attributes in the extent and/or intent of a formal concept in the latent space of factors, the more important the formal concept is considered to be. The presented approach is intuitive and easily explainable for users. It can be used together with other approaches to rank formal concepts.

2. Related Work

The reduction of the number of formal concepts and, thus, the size of concept lattices can be accomplished directly (removing formal concepts that do not satisfy a requirement) or in an indirect way (through handling formal contexts).

The aim of the approach in [5] is to find a decomposition of a Boolean (binary) matrix (formal context) with the smallest number of factors (that correspond to formal concepts) as possible. These factor concepts can be considered more important than other concepts of the formal context.

The usage of rank-k SVD in order to reduce the size of the corresponding concept lattice is proposed in [8]. However, SVD is not used to reduce the number of objects and/or attributes, but instead, to remove noise in an input table. Subsequently, the number of formal concepts is reduced. In [6], SVD is used to decompose a document-term matrix into a much smaller matrix where terms are related to a set of dimensions (factors) instead of documents. This term-dimension matrix is then converted into a binary matrix using a probabilistic approach.

The main idea of the JBOS (junction based on object similarity) method is that groups of similar objects are replaced by representative ones. The similarity of two objects is the sum of the weights of attributes in which the objects agree with each other (both objects have them or both do not have them) [7].

Another way is to reduce the number of formal concepts by means of attribute-dependency formulas (ADF) expressing the relative importance of attributes [3]. ADF depend on the purpose and have to be specified by an expert. Only formal concepts satisfying the set of ADF are relevant. The approach in [2] also utilizes background knowledge. After a user assigns weights to attributes, values of formal concepts are determined. Formal concepts with higher values are considered more important.

The idea of basic level of concepts appeared in [4]. Concepts in the basic level represent those concepts which are preferred by humans to use when describing the world around. The cohesion of a formal concept defined in [4], unlike the coherence proposed in this work, is a measure of whether the objects in its extent are pairwise similar.

The notion of the stability of a formal concept was introduced in [12]. The stability index indicates how much the intent of a concept depends on the set of objects in the extent (intentional stability). Extentional stability was defined analogously. Two other indices, probability and separation, are proposed in [11] and their performance on noisy data is discussed.

Another option to reduce the size of concept lattices is to consider only frequent formal concepts for a user given minimum support (Iceberg concept lattice) [15]. Note that a concept (A, B) is frequent if the fraction of objects that contain the attributes in B is above the minimum support threshold.

3. Formal Concept Analysis

A *formal context* is a triple (X, Y, R) consisting of a set $X = \{x_1, \ldots, x_n\}$ of objects, a set $Y = \{y_1, \ldots, y_m\}$ of attributes and a binary relation $R \subseteq X \times Y$ between them. We write $(x, y) \in R$ if the object x has the attribute y.

For a set $A \subseteq X$ of objects and a set $B \subseteq Y$ of attributes we define $A' = \{y \in Y : (\forall x \in A)(x, y) \in R\}$ and $B' = \{x \in X : (\forall y \in B)(x, y) \in R\}$. A' is the set of attributes common to the objects in A and B' is the set of objects which have all the attributes in B.

A *formal concept* of (X, Y, R) is a pair (A, B) where $A \subseteq X, B \subseteq Y, A' = B$ and $B' = A$. A and B are called the *extent* and the *intent* of the concept (A, B), respectively. The set of all concepts of (X, Y, R) is denoted by $\mathcal{B}(X, Y, R)$. $A \subseteq X$ $(B \subseteq Y)$ is an extent (intent) if and only if $A'' = A$ $(B'' = B)$.

We define a partial order $\leq$ on $\mathcal{B}(X, Y, R)$ by $(A_1, B_1) \leq (A_2, B_2)) \Leftrightarrow A_1 \subseteq A_2$ (equivalently, $B_1 \supseteq B_2$). The set of all concepts of (X, Y, R) ordered by $\leq$ constitutes the *concept lattice* $(\mathcal{B}(X, Y, R), \leq)$ of (X, Y, R) [16].

For more details on Formal Concept Analysis we refer to [9].

Example: The formal context in Table 1 induces 26 formal concepts:
$C_1 = (\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}, \emptyset)$,
$C_2 = (\{1, 2, 3, 5, 6, 7, 10, 12, 14\}, \{1\})$, $C_3 = (\{1, 9, 10, 11\}, \{4\})$,
$C_4 = (\{4, 8, 9, 10, 11, 13, 14, 15\}, \{6\})$,
$C_5 = (\{1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 15\}, \{8\})$,
$C_6 = (\{1, 10\}, \{1, 4\})$, $C_7 = (\{9, 10, 11\}, \{4, 6\})$,
$C_8 = (\{6, 8, 12\}, \{5, 8\})$, $C_9 = (\{4, 8, 9, 11, 13, 15\}, \{6, 8\})$,
$C_{10} = (\{8, 10, 13, 14, 15\}, \{6, 10\})$, $C_{11} = (\{1, 2, 3, 4, 5, 6, 7, 9, 11, 12\}, \{8, 9\})$,
$C_{12} = (\{10, 14\}, \{1, 6, 10\})$, $C_{13} = (\{1, 9, 11\}, \{4, 8, 9\})$,

TABLE 1. An illustrative example of a formal context of animals and their attributes (a cross in a row x and a column y indicates that the object x has the attribute y)

		1	2	3	4	5	6	7	8	9	10	11
		fur (hair)	feathers	scales	can fly	lives in water	lays eggs	produces milk	has a backbone	warm-blooded	cold-blooded	domestic
1	Bat	×			×			×	×	×		
2	Bear	×						×	×	×		
3	Cat	×						×	×	×		×
4	Chicken		×				×		×	×		×
5	Dog	×						×	×	×		×
6	Dolphin	×				×		×	×	×		
7	Elephant	×						×	×	×		
8	Frog					×	×		×		×	
9	Hawk		×		×		×		×	×		
10	Housefly	×			×		×				×	
11	Owl		×		×		×		×	×		
12	Sea lion	×				×		×	×	×		
13	Snake			×			×		×		×	
14	Spider	×					×				×	
15	Turtle			×			×		×		×	

$C_{14} = (\{8, 13, 15\}, \{6, 8, 10\})$, $C_{15} = (\{3, 4, 5\}, \{8, 9, 11\})$,
$C_{16} = (\{10\}, \{1, 4, 6, 10\})$, $C_{17} = (\{1, 2, 3, 5, 6, 7, 12\}, \{1, 7, 8, 9\})$,
$C_{18} = (\{4, 9, 11\}, \{2, 6, 8, 9\})$, $C_{19} = (\{13, 15\}, \{3, 6, 8, 10\})$,
$C_{20} = (\{8\}, \{5, 6, 8, 10\})$, $C_{21} = (\{1\}, \{1, 4, 7, 8, 9\})$,
$C_{22} = (\{6, 12\}, \{1, 5, 7, 8, 9\})$, $C_{23} = (\{3, 5\}, \{1, 7, 8, 9, 11\})$,
$C_{24} = (\{9, 11\}, \{2, 4, 6, 8, 9\})$, $C_{25} = (\{4\}, \{2, 6, 8, 9, 11\})$,
$C_{26} = (\emptyset, \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\})$

4. Outline of Our Approach

Each formal context (input data for FCA) can be viewed as a matrix with n rows representing objects, m columns representing attributes and values 1 or 0 depending on whether an object has a given attribute or not. Thence, we can refer to a context as a matrix $R \in \{0, 1\}^{n \times m}$.

Consider a formal context R in Table 1, in which objects $x_1, \ldots, x_n \in X$ are animals and $y_1, \ldots, y_m \in Y$ are attributes which relate to animals, e.g. can fly, has a backbone, is warm-blooded, etc. Using a matrix factorization method we can create an approximation of a formal context R by a product of two or

more matrices. Factorizing R means mapping the objects and attributes to a common k-dimensional latent space, the coordinates of which are called the factors. For attributes of animals, the discovered factors might measure obvious dimensions such as whether an animal is a mammal or whether an animal can fly; less well-defined dimensions such as whether an animal is dangerous or not; or, completely uninterpretable dimensions. For animals, each factor measures the extent to which the animal possesses the corresponding factor. Note that we are not concerned in the exact interpretation of the factors in this work since it belongs rather to areas of human sciences (psychology, sociology, etc.).

We use the idea of mapping of objects and attributes to a common latent factor space to define the coherence of a formal concept. The coherence is based on the distance between objects and/or attributes in the common latent factor space; objects which are close to each other share more common characteristics than objects which are remote from each other (similarly for attributes). For example, the distance between cat and dog should be small unlike the distance between cat and housefly. The attributes cold-blooded and warm-blooded should be remote from each other since these attributes exclude each other, i.e. if an animal is cold-blooded, it can not be warm-blooded and vice versa. Naturally, the location of objects and attributes in a latent factor space is dependent on an input (formal context) what will be seen later in Section 6.

5. Matrix Factorization

For the decomposition of formal contexts (Boolean matrices), Boolean Matrix Factorization is a natural choice. However, we also provided experiments with other factorization techniques, namely Singular Value Decomposition and Non-negative Matrix Factorization.

5.1. **Boolean Matrix Factorization (BMF).** The aim of Boolean Matrix Factorization (BMF) is to find a decomposition of a given matrix $X \in \{0,1\}^{n\times m}$ into a Boolean matrix product

$$X = A \circ B \quad (\text{or } X \approx A \circ B)$$

of matrices $A \in \{0,1\}^{n\times k}$ and $B \in \{0,1\}^{k\times m}$. [5]

A Boolean matrix product $A \circ B$ is defined by

$$(A \circ B)_{ij} = \max_{l=1}^{k} A_{il} \cdot B_{lj},$$

where max denotes the maximum and $\cdot$ the ordinary product.

A decomposition of $X = A \circ B$ corresponds to a discovery of k factors that exactly or approximately explain the data. The least k for which an exact

decomposition $X = A \circ B$ exists is called the *Boolean rank* (Schein rank) of X.

There are two different problems to solve in BMF:

- Discrete Basis Problem (DBP): Given $X \in \{0,1\}^{n \times m}$ and an integer $k > 0$, find $A \in \{0,1\}^{n \times k}$ and $B \in \{0,1\}^{k \times m}$ that minimize $||X - A \circ B|| = \sum_{ij} |X_{ij} - (A \circ B)_{ij}|$. [14]
- Approximate Factorization Problem (AFP): Given X and an error $\varepsilon \geq 0$, find $A \in \{0,1\}^{n \times k}$ and $B \in \{0,1\}^{k \times m}$ with k as small as possible such that $||X - A \circ B|| \leq \varepsilon$. [5]

In this paper, we have used the greedy approximation algorithm for BMF described in [5] (where it is called Algorithm 2).

5.2. **Singular Value Decomposition (SVD).** Singular Value Decomposition (SVD) is a factorization of a matrix $X \in \mathbb{R}^{n \times m}$ by the product of three matrices $X = U\Sigma V^T$ where $U \in \mathbb{R}^{n \times n}$, $\Sigma \in \mathbb{R}^{n \times m}$ and $V \in \mathbb{R}^{m \times m}$ such that $U^T U = I, V^T V = I$ (where I is an identity matrix), column vectors of U (left-singular vectors) are orthonormal eigenvectors of XX^T, column vectors of V (right-singular vectors) are orthonormal eigenvectors of $X^T X$ and Σ contains singular values of X at the diagonal in descending order.

We can create an approximation $\hat{X}$ of a matrix X as

$$X \approx \hat{X} = U\Sigma V^T,$$

where $U \in \mathbb{R}^{n \times k}$, $\Sigma \in \mathbb{R}^{k \times k}$ and $V \in \mathbb{R}^{m \times k}$.

5.3. **Non-negative Matrix Factorization (NMF).** Let X be an $n \times m$ non-negative matrix and $k > 0$ an integer. The goal of Non-negative Matrix Factorization (NMF) [13] is to find an approximation

$$X \approx WH,$$

where W and H are $n \times k$ and $k \times m$ non-negative matrices, respectively.

The matrices W and H are estimated by minimizing the function

$$D(X, WH) + Reg(W, H),$$

where D measures the divergence and Reg is an optional regularization function. The different types of NMF arise from using different cost functions for measuring the divergence between X and WH, and by regularization of W and/or H.

The quality of the approximation is quantified by a cost function D. The common cost function between two non-negative matrices A and B is the squared error (Frobenius norm)

$$D(A, B) = \sum_{ij} (a_{ij} - b_{ij})^2.$$

6. The Proposed Approach

The notation we use in this and subsequent sections is the following:

- $distance(x_1, x_2)$ denotes a distance between objects x_1 and x_2 in the latent space,
- $coherence(C)$ denotes the degree of coherence of a formal concept C

Let R be a formal context, $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ be the sets of objects and attributes of R, respectively. After the decomposition of R each object $x_i \in X$ is represented by a k-dimensional vector of latent factors $(x_{i_1}, \ldots, x_{i_k})$ describing the object and each attribute $y_j \in Y$ is represented by a k-dimensional vector of factors $(y_{j_1}, \ldots, y_{j_k})$ describing the attribute. Obviously, some objects are close to each other, while other objects are far away from each other (depending on the distance between objects) in the space of factors.

In our experiments, we have used the Manhattan distance (L_1 distance) and the Euclidean distance (Euclidean metric, L_2 distance). The distance between two objects $x_1, x_2 \in X$ (i.e. k-dimensional vectors $(x_{1_1}, \ldots, x_{1_k})$ and $(x_{2_1}, \ldots, x_{2_k})$ of latent factors) is given by

(Manhattan distance) $$distance(x_1, x_2) = \frac{1}{k} \sum_{l=1}^{k} |x_{1_l} - x_{2_l}|$$

(Euclidean distance) $$distance(x_1, x_2) = \sqrt{\frac{1}{k} \sum_{l=1}^{k} (x_{1_l} - x_{2_l})^2}$$

To have $distance \in [0, 1]$ we put $\frac{1}{k}$ to the equations Manhattan distance and Euclidean distance. The distance between attributes or between an object and an attribute in the latent factor space can be computed similarly.

One natural way to measure the coherence of a formal concept is by using the distance (Manhattan, Euclidean) between the objects in the extent of the formal concept as

$$coherence_X^{\max}(A, B) = 1 - \max_{x_1, x_2 \in A} distance(x_1, x_2) \tag{1}$$

Alternatively, we might put

$$coherence_X^{\text{avg}}(A, B) = 1 - \frac{\sum_{\{x_1, x_2\} \subseteq A, x_1 \neq x_2} distance(x_1, x_2)}{|A|(|A| - 1)/2} \tag{2}$$

Simply, $coherence_X^{\max}(A, B)$ is computed by the maximum distance between any two objects in the extent of (A, B) and $coherence_X^{\text{avg}}(A, B)$ is computed using the average distance between two objects in the extent of (A, B). Thus, $coherence_X^{\max}(A, B) \leq coherence_X^{\text{avg}}(A, B)$.

Formal concepts with similar objects (i.e. objects that share many common attributes) in their extents have a high degree of $coherence_X^{\max}$ and $coherence_X^{\text{avg}}$ (provided that similar objects are close to each other while distinct objects are remote from each other in the space of factors what will be seen later).

Similarly, the coherence of a formal concept can be measured using the distance between the attributes in its intent (denoted by $coherence_Y^{\max}$ and $coherence_Y^{\text{avg}}$). Alternatively, we can use the distance between both, the objects and attributes in the extent and intent of a formal concept, respectively (denoted by $coherence^{\max}$ and $coherence^{\text{avg}}$).

It is easy to see that if $(A_1, B_1) \leq (A_2, B_2)$, then $coherence_X^{\max}(A_1, B_1) \geq coherence_X^{\max}(A_2, B_2)$ and $coherence_Y^{\max}(A_1, B_1) \leq coherence_Y^{\max}(A_2, B_2)$.

Remark: From the above mentioned assumptions it follows that the decision of whether to use $coherence_Y^?$ or $coherence_X^?$, where ? = max or ? = avg, depends on user/expert preferences. $coherence_Y^?$ prefers specific concepts (a concept is specific if it consists of a few objects that share many attributes, see Fig. 2) while $coherence_X^?$ tends to prefer general concepts (a concept is general if it consists of many objects that have only a few attributes in common, see Fig. 3).

Now, we are able to assign a degree of coherence to each formal concept of a formal context. We consider formal concepts with higher degrees of coherence more important.

6.1. **Illustrative Example.** In this section we demonstrate our approach on a small example. It depends on the outcome of a matrix factorization method whether the results provided by our approach will be good or not. Therefore, we first address the problem of matrix factorization, and then we analyze the results themselves.

For the decomposition of the formal context in Table 1 we utilize Boolean Matrix Factorization (BMF) due to the good interpretability of factors. Using BMF the animals and their attributes are mapped to 8-dimensional space of latent factors which is shown in Fig. 1.

The interpretation of the factors might be the following: The first factor can be interpreted as the property of being a mammal (manifestations of the factor are: *fur (hair)*, *produces milk*, *has a backbone*, *warm-blooded*) and the second factor can be interpreted as the property of being a bird (manifestations of this factor are *feathers*, *lays eggs*, *has a backbone*, *warm-blooded*). The other factors relates to the attributes *cold-blooded*, *scales*, *can fly*, *lives in water*, *domestic*, *fur (hair)*, respectively.

The animals *bear* and *elephant* are mapped to the same point in the space of latent factors. The same is true for *cat* and *dog*, *dolphin* and *sea lion*, *hawk*

FIGURE 1. The decomposition of the formal context in Table 1 using BMF

		factor 1	factor 2	factor 3	factor 4	factor 5	factor 6	factor 7	factor 8
1	Bat	×				×			×
2	Bear	×							×
3	Cat	×						×	×
4	Chicken		×					×	
5	Dog	×						×	×
6	Dolphin	×					×		×
7	Elephant	×							×
8	Frog			×			×		
9	Hawk		×			×			
10	Housefly			×		×			×
11	Owl		×			×			
12	Sea lion	×					×		×
13	Snake			×	×				
14	Spider			×					×
15	Turtle			×	×				

	fur (hair)	feathers	scales	can fly	lives in water	lays eggs	produces milk	has a backbone	warm-blooded	cold-blooded	domestic
factor 1	×						×	×	×		
factor 2		×				×		×	×		
factor 3						×				×	
factor 4			×			×		×		×	
factor 5				×							
factor 6					×			×			
factor 7								×	×		×
factor 8	×										

and *owl* as well as *snake* and *turtle*. According to Table 1 these animals agree with each other, i.e. either all of the animals have some attribute or none of them. In the contrary, for the animals *owl* and *sea lion*, if one of the animals has a factor, then the second one does not have the factor. Correspondingly, if *owl* has an attribute in the formal context in Table 1, then *sea lion* does not have the attribute and vice versa (except for the attributes *has a backbone* and *warm-blooded* contained in the manifestations in both of the first two factors).

In the space of factors, the attributes *lays eggs* and *cold-blooded* differ only in the second factor. All the animals in the formal context in Table 1 except for *chicken*, *hawk* and *owl* (e.g. except for birds) agree on these attributes.

After the decomposition of the formal context in Table 1, we can measure the distance between animals (objects) and/or their properties (attributes) in the space of factors. Using Manhattan distance, $distance(bear, elephant) = 0$, $distance(owl, sea\ lion) = \frac{5}{8}$, $distance(lays\ eggs, cold\text{-}blooded) = \frac{1}{8}$.

It is important to notice that the location of objects and attributes in the common factor space depends on the formal context, mainly on the selection of appropriate attributes. A formal context that does not contain "good" attributes may cause that different animals will have many factors in common.

For each formal concept of the formal context in Table 1 we can compute the degree of coherence. For example,
$coherence_X^{\max}(C_{18}) = 1 - \frac{2}{8} = 0.75$,
$coherence_X^{\text{avg}}(C_{18}) = 1 - \frac{\frac{2}{8}+\frac{2}{8}+0}{3} = 1 - \frac{1}{6} = \frac{5}{6}$,
$coherence_Y^{\max}(C_{18}) = 1 - \frac{4}{8} = 0.5$,
$coherence_Y^{\text{avg}}(C_{18}) = 1 - \frac{\frac{2}{8}+\frac{4}{8}+\frac{2}{8}+\frac{4}{8}+\frac{4}{8}+\frac{2}{8}}{6} = 1 - \frac{3}{8} = \frac{5}{8}$.

Formal concepts $C_1, \ldots, C_{26}$ and their degrees of coherence are shown in Table 2. Remember that the higher the coherence, we consider the formal concept more important.

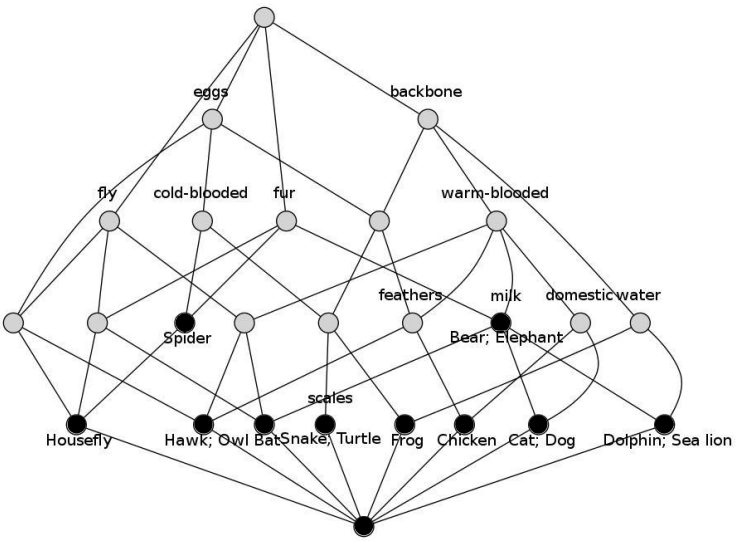


FIGURE 2. The concept lattice of animals. The formal concepts with $coherence_X^{\text{avg}}$ greater or equal to 0.85 using BMF are highlighted in black

TABLE 2. The coherence (computed using Manhattan distance) of the formal concepts of Table 1 rounded to two decimal places

	intent of (A, B)											coherence of (A, B)					
	fur (hair)	feathers	scales	can fly	lives in water	lays eggs	produces milk	has a backbone	warm-blooded	cold-blooded	domestic	$coherence^{\max}$	$coherence^{\text{avg}}$	$coherence_X^{\max}$	$coherence_X^{\text{avg}}$	$coherence_Y^{\max}$	$coherence_Y^{\text{avg}}$
1.												0.38	0.60	0.38	0.60	1.00	1.00
2.	×											0.50	0.78	0.50	0.76	1.00	1.00
3.				×								0.63	0.75	0.63	0.71	1.00	1.00
4.						×						0.38	0.64	0.38	0.63	1.00	1.00
5.								×				0.25	0.57	0.38	0.59	1.00	1.00
6.	×			×								0.63	0.73	0.75	0.75	0.63	0.63
7.				×		×						0.50	0.70	0.63	0.75	0.50	0.50
8.					×			×				0.38	0.65	0.63	0.75	0.50	0.50
9.						×		×				0.38	0.60	0.50	0.63	0.50	0.50
10.						×				×		0.50	0.76	0.63	0.75	0.88	0.88
11.								×	×			0.25	0.63	0.38	0.66	0.75	0.75
12.	×					×				×		0.38	0.65	0.88	0.88	0.38	0.58
13.				×				×	×			0.25	0.60	0.63	0.75	0.25	0.50
14.						×		×		×		0.38	0.70	0.75	0.83	0.38	0.58
15.								×	×		×	0.50	0.71	0.63	0.75	0.50	0.67
16.	×			×		×				×		0.38	0.60	1.00	1.00	0.38	0.58
17.	×						×	×	×			0.25	0.74	0.75	0.85	0.38	0.65
18.		×				×		×	×			0.38	0.68	0.75	0.83	0.50	0.63
19.			×			×		×		×		0.38	0.74	1.00	1.00	0.38	0.65
20.					×	×		×		×		0.38	0.60	1.00	1.00	0.38	0.56
21.	×			×			×	×	×			0.25	0.61	1.00	1.00	0.25	0.60
22.	×				×		×	×	×			0.38	0.67	1.00	1.00	0.38	0.63
23.	×						×	×	×		×	0.38	0.70	1.00	1.00	0.38	0.65
24.		×		×		×		×	×			0.25	0.64	1.00	1.00	0.25	0.58
25.		×				×		×	×		×	0.50	0.68	1.00	1.00	0.50	0.63
26.	×	×	×	×	×	×	×	×	×	×	×	0.25	0.63	1.00	1.00	0.25	0.63

The most coherent formal concepts using $coherence_X^{\text{avg}}$ (the 4th column in Table 2) sorted in descending degree are C_{16}, $C_{19} - C_{25}$, C_{12} and C_{17} (Fig. 2). The formal concepts C_{19}, C_{22}, C_{23}, C_{12} and C_{17} can be named as "reptiles" (*snake*, *turtle*), "sea mammals" (*dolphin*, *sea lion*), "pets" (*cat*, *dog*), "invertebrate animals" (*housefly*, *spider*) and "mammals" (*bat*, *bear*, *cat*, *dog*, *dolphin*, *elephant*, *sea lion*), respectively.

Next, consider the most coherent concepts using the coherence measured on the attributes only $coherence_Y^{\max}$ (the 5th column in Table 2). The concepts in descending order of the degree of coherence are $C_2 - C_5$, C_{10}, C_{11}, C_6, C_7, C_8, C_9, C_{15}, C_{18} and C_{25} (Fig. 3). The concepts C_5, C_{10}, C_{11}, C_8, C_{15} and

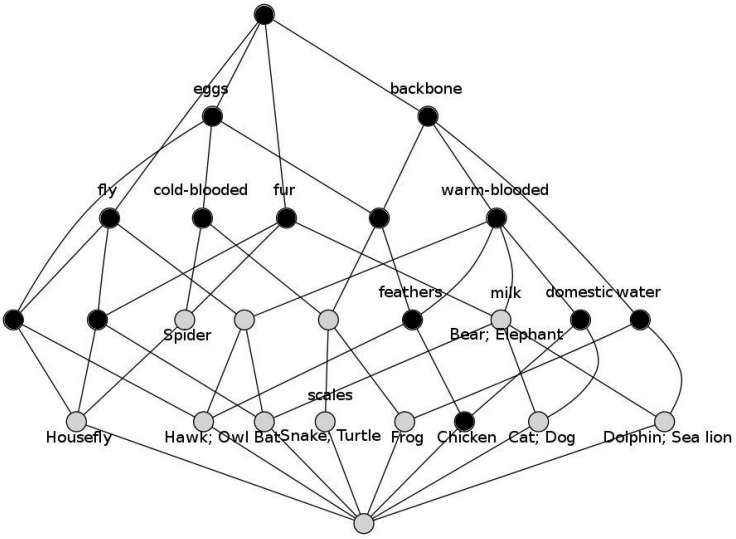


FIGURE 3. The concept lattice of animals. The formal concepts having $coherence_Y^{\max}$ greater or equal to 0.75 using BMF are highlighted in black

C_{18} represent "vertebrate animals", "cold-blooded animals", "warm-blooded animals", "aquatic animals", "domestic animals" and "birds".

The interpretation of other results (i.e. these provided by $coherence^{\max}$, $coherence^{\text{avg}}$, $coherence_X^{\max}$ and $coherence_Y^{\text{avg}}$) is left to the reader.

The decision of whether to compute the coherence on objects or attributes as well as the use of $coherence^{\max}$ or $coherence^{\text{avg}}$ depends on the purpose of the concrete application of FCA on the data and also on other user-related factors.

7. Experiments

In this section, we present some experiments we have performed to give a deeper insight into the behaviour of the proposed method for ranking formal concepts. Benchmark data sets used for these experiments are taken from the UCI Machine Learning Repository [1] characteristics of which are shown in Table 3.

7.1. **Experiment 1.** In the first experiment, we have explored the degrees of coherence that are assigned to formal concepts.

Using Boolean Matrix Factorization (BMF) for the decomposition of formal contexts we have found out that (see Table 4):

TABLE 3. Some characteristics of the used data sets in our experiments

Dataset	# Objects	# Attributes	# Factors (BMF)	# Formal Concepts
Car	1728	25	25	12640
Spect Heart	267	46	46	2135549
Tic-tac-toe	958	29	29	59505
Wine	178	68	57	24423

- Many concepts of the formal contexts (data sets) have the same degree of coherence if we measure the coherence using maximum distance between objects and/or attributes in the factor space. For example, for wine data set the total number of formal concepts is 24423, each of which is assigned 1 out of 12 degrees of coherence (using $coherence^{\max}$).
- The number of distinct coherence values computed by the maximum distance is identical using either of the two distance measures (Manhattan, Euclidean).
- $coherence_X^{\max}$ and $coherence_X^{\text{avg}}$ provide more distinct coherence values than $coherence_Y^{\max}$ and $coherence_Y^{\text{avg}}$, respectively. It follows from the fact that the number of objects is greater than the number of attributes for each data set.
- It is appropriate to utilize the Euclidean distance instead of the Manhattan distance for measuring the coherence, because the number of distinct degrees of coherence that are assigned to formal concepts is greater if we use the Euclidean distance (what is not surprising, since the factor matrices are binary, i.e. contain only 0s and 1s).
- For tic-tac-toe data the number of distinct degrees of coherence assigned to formal concepts using $coherence_Y^{\max}$ and $coherence_Y^{\text{avg}}$ is very small, because each attribute possesses a unique factor no other attribute has.

We can conclude that, using BMF, the same coherence degree is assigned to many formal concepts (see Table 4). These concepts are then ranked at the same position which is not useful for a user.

We have also carried out similar experiment where SVD and NMF were used for the decomposition of formal contexts. Remember that factor matrices generated by SVD and NMF are real-valued matrices. Therefore, using the average distance between objects and/or attributes in a factor space, almost all formal concepts take on different degrees of coherence. Further, using the maximum distance between objects and/or attributes in a space of factors, the number of distinct degrees of coherence is greater (in comparison to the case of using BMF) when we use SVD or NMF for the decomposition of formal contexts.

TABLE 4. The numbers of distinct values of coherence that the formal concepts of the formal contexts (data sets) take on using the respective ways of measuring the coherence (BMF was used for decomposition of data sets)

	Manhattan distance						Euclidean distance					
Dataset	$coherence^{\max}$	$coherence^{\text{avg}}$	$coherence_X^{\max}$	$coherence_X^{\text{avg}}$	$coherence_Y^{\max}$	$coherence_Y^{\text{avg}}$	$coherence^{\max}$	$coherence^{\text{avg}}$	$coherence_X^{\max}$	$coherence_X^{\text{avg}}$	$coherence_Y^{\max}$	$coherence_Y^{\text{avg}}$
Car	7	826	8	609	6	28	7	2326	8	1381	6	57
Spect Heart	15	231041	25	153976	5	127	15	2072196	25	1969571	5	372
Tic-tac-toe	8	1156	10	1048	2	2	8	6557	10	5402	2	6
Wine	12	4906	16	3127	17	651	12	24360	16	16868	17	8132

The use of SVD and NMF in our approach allow us to differentiate better between formal concepts with respect to degrees of coherence, and thus are better for ranking formal concepts according to their coherence. From this point of view, it is also appropriate to measure the coherence utilizing the average distance (not the maximum distance) between objects and/or attributes in the extents and/or intents of formal concepts, respectively.

7.2. **Experiment 2.** The aim of the second experiment is to investigate the impact of matrix factorization methods on the selection of important (coherent) formal concepts. Hence, we have compared the top-k most coherent formal concepts resulting from our approach by using various methods of matrix decomposition.

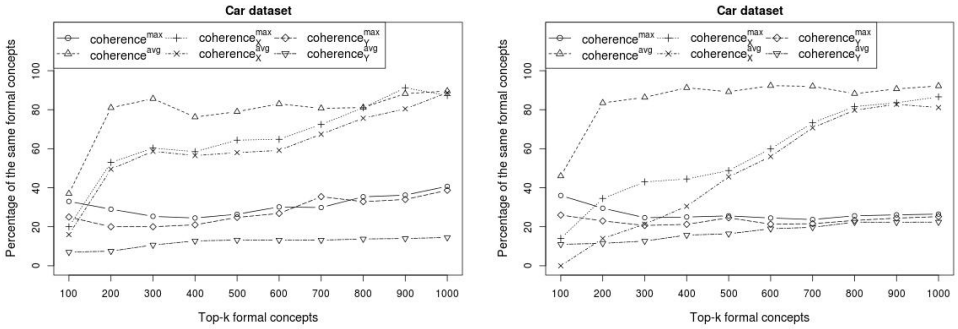
Based on the conclusions of the previous experiment, matrix factorization methods that decompose a matrix into a product of real-valued matrices are better if we want to rank formal concepts according to their degrees of coherence. Thus, in this experiment we have used SVD and NMF for matrix decomposition.

For a formal concept (A, B) it holds that if $|A| \leq 1$ ($|B| \leq 1$), then $coherence_X^{?} = 1$ ($coherence_Y^{?} = 1$), where $? = \max$ or $? = \text{avg}$. The number of formal concepts satisfying these conditions, and thus having the corresponding degrees of coherence equal to 1 are shown in Table 5. Since this assertion holds regardless of the selected method of matrix factorization, we did not consider such formal concepts in this experiment. Obviously, formal concepts that do not satisfy these conditions can also have degrees of coherence equal to 1.

The results of the comparison of the top-k most coherent formal concepts using SVD and NMF are shown in Fig. 4 – Fig. 7.

TABLE 5. The number of formal concepts (A, B) such that $|A| \leq 1$ or $|B| \leq 1$

Dataset	# Formal Concepts (A, B) having $\lvert A\rvert \leq 1$	# Formal Concepts (A, B) having $\lvert B\rvert \leq 1$
Car	1729	23
Spect Hear	215	44
Tic-tac-toe	959	30
Wine	169	37



(a) Manhattan distance

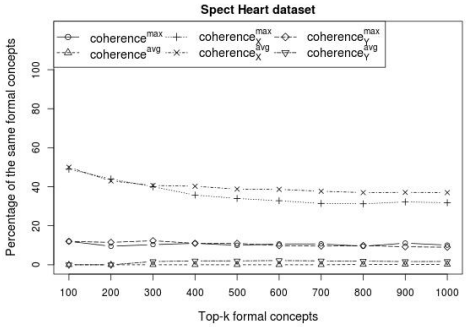
(b) Euclidean distance

FIGURE 4. The percentage of the same formal concepts from the top-k formal concepts using SVD and NMF for the decomposition of Car dataset using Manhattan distance (Fig. 4(a)) and Euclidean distance (Fig. 4(b)).

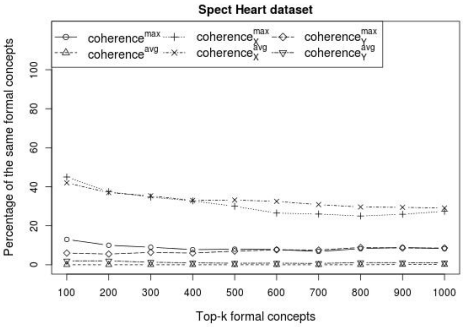
The used matrix factorization method has only a little influence on formal concepts provided by $coherence^{\text{avg}}$ (except for the Spect Heart dataset). Approximately 80% of formal concepts provided by $coherence^{\text{avg}}$ are the same if we decompose data sets using SVD or NMF (for the Car and Tic-tac-toe datasets). On the other hand, $coherence^{\text{max}}$, $coherence_Y^{\text{max}}$ and $coherence_Y^{\text{avg}}$ are quite sensitive on the selected method of matrix decomposition. The results are similar regardless of the computation of the distance in a space of factors (Manhattan distance, Euclidean distance).

8. CONCLUSIONS

We have introduced a novel approach to rank (and thus to reduce the number of) formal concepts utilizing different types of matrix factorization methods. Besides the intuitive choice, the Boolean Matrix Factorization technique (BMF), we have utilized also Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF). As our experiments showed, using

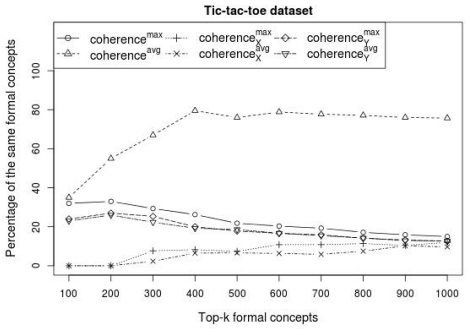


(a) Manhattan distance

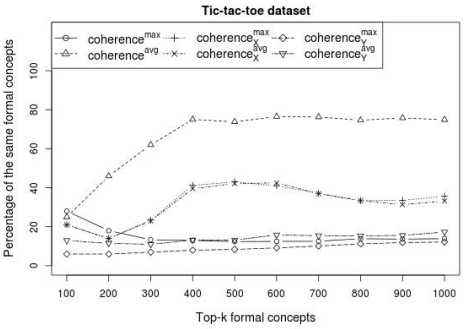


(b) Euclidean distance

FIGURE 5. The percentage of the same formal concepts from the top-k formal concepts using SVD and NMF for the decomposition of Spect Heart dataset using Manhattan distance (Fig. 5(a)) and Euclidean distance (Fig. 5(b)).



(a) Manhattan distance

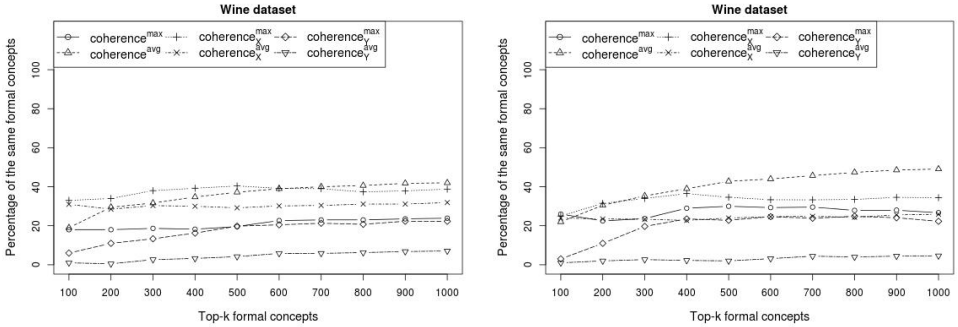


(b) Euclidean distance

FIGURE 6. The percentage of the same formal concepts from the top-k formal concepts using SVD and NMF for the decomposition of Tic-tac-toe dataset using Manhattan distance (Fig. 6(a)) and Euclidean distance (Fig. 6(b)).

BMF in our approach results in a case when only a small number of distinct values are assigned to formal concepts and thus many formal concepts have the same degree of coherence which is not helpful in ranking. However, having just a small number of different ranking degrees could be interesting in some cases of application of FCA to data.

The main research issue we would like to focus on the following issues:



(a) Manhattan distance

(b) Euclidean distance

FIGURE 7. The percentage of the same formal concepts from the top-k formal concepts using SVD and NMF for the decomposition of Wine dataset using Manhattan distance (Fig. 7(a)) and Euclidean distance (Fig. 7(b)).

- Experimental evaluation of the proposed approach on several real-world data sets including qualitative evaluation of the results by domain experts.
- Comparison with other techniques to select important formal concepts, in particular with the one for selecting basic level concepts [4].

Acknowledgements: This publication is the result of the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF and VEGA 1/0832/12.

REFERENCES

[1] K. Bache, M. Lichman, *UCI Machine Learning Repository*. 2013. URL http://archive.ics.uci.edu/ml.

[2] R. Bělohlávek, J. Macko, *Selecting Important Concepts Using Weights*. International Conference on Formal Concept Analysis, LNAI, vol. 6628, Springer, 2011, pp. 65-80.

[3] R. Bělohlávek, V. Sklenář, *Formal Concept Analysis Constrained by Attribute-Dependency Formulas*, International Conference on Formal Concept Analysis, LNAI, vol. 3403, Springer, 2005, pp. 176-191.

[4] R. Bělohlávek, M. Trnečka, *Basic Level of Concepts in Formal Concept Analysis*. International Conference on Formal Concept Analysis, LNAI, vol. 7278, Springer, 2012, pp. 28-44.

[5] R. Bělohlávek, V. Vychodil, *Discovery of optimal factors in binary data via a novel method of matrix decomposition*. Journal of Computer and System Sciences, vol. 76, 2010, pp. 3-20.

[6] V. Codocedo, C. Taramasco, H. Astudillo, *Cheating to achieve Formal Concept Analysis over a large formal context.* Concept Lattices and Their Application, 2011, pp. 349-362.
[7] S. M. Dias, N. J. Vieira, *Reducing the Size of Concept Lattices: The JBOS approach.* Concept Lattices and Their Application, CEUR WS, vol. 672, 2010, pp. 80-91.
[8] P. Gajdoš, P. Moravec, V. Snášel, *Concept Lattice Generation by Singular Value Decomposition.* Concept Lattices and Their Application, 2004, pp. 102-110.
[9] B. Ganter, R. Wille, *Formal concept analysis: Mathematical foundations.* Springer, 1999.
[10] J. Han, H. Cheng, D. Xin, X. Yan, *Frequent pattern mining: current status and future directions.* Data Mining and Knowledge Discovery, vol. 15, 2007, pp. 55-86.
[11] M. Klimushkin, S. Obiedkov, C. Roth, *Approaches to the Selection of Relevant Concepts in the Case of Noisy Data.* International Conference on Formal Concept Analysis, LNAI, vol. 5986, Springer, 2010, pp. 255-266.
[12] S. O. Kuznetsov, *On stability of a formal concept.* Annals of Mathematics and Artificial Intelligence, vol. 49(1-4), 2007, pp. 101-115.
[13] D. D. Lee, H. S. Seung, *Algorithms for non-negative matrix factorization.* Advances in neural information processing systems, vol. 13, 2001.
[14] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, H. Mannila, *The Discrete Basis Problem.* IEEE Transactions on Knowledge and Data Engineering, vol. 20(10), 2008, pp. 1348-1362.
[15] G. Stumme, *Efficient Data Mining Based on Formal Concept Analysis.* DEXA, Springer, LNCS, vol. 2453, 2002, pp. 534-546.
[16] R. Wille, *Restructuring lattice theory: An approach based on hierarchies of concepts.* Ordered Sets, vol. 83, 1982, pp. 445-470.

Institute of Computer Science, Faculty of Science, Pavol Jozef Šafárik University, Jesenná 5, Košice, Slovakia
E-mail address: {lenka.piskova, tomas.horvath, stanislav.krajci}@upjs.sk

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Special Issue 2, 2014
ICFCA 2014: 12th International Conference on Formal Concept Analysis, Cluj-Napoca, June 10-13, 2014

TRIADIC APPROACH TO CONCEPTUAL DESIGN OF XML DATA

CHRISTIAN SĂCĂREA AND VIORICA VARGA

Abstract. As XML becomes a popular data representation and exchange format over the web, XML schema design has become an important research area. Discovering XML data redundancies from the data itself becomes necessary and it is an integral part of the schema refinement (or re-design) process. Different authors present the notion of functional dependency in XML data and normal forms for XML data. Yu and Yagadish (2008) give the definition of the Generalized Tree Tuple (GTT) and XNF normal form. They present also a hierarchical and a flat representation of XML data. The hierarchical representation of XML data from the paper of Yu and Yagadish (2008) is used to define a triadic FCA approach for a conceptual model of XML data. The formal tricontext of functional dependencies with respect to a tuple class is given.

1. Introduction and Previous Work

The goal of relational database design is to generate a set of relation schemas that allows us to store information without unnecessary redundancy. The relation scheme obtained by translating the Entity-Relationship model is a good starting point, but we still need to develop new techniques to detect possible redundancies in the preliminary relation scheme. The normal form satisfied by a relation is a measure of the redundancy in the relation. In order to analyze the normal form of a relation we need to detect the functional dependencies that are present in the relation.

XML is a popular data representation and exchange format over the web. XML data design must ensure that there are no unintended redundancies, since these can generate data inflation and transfer costs, as well as unnecessary storage. Hence, the good design of XML schemas is an important issue. Redundancies are captured as functional dependencies in relational databases

Received by the editors: March 25, 2014.

2010 *Mathematics Subject Classification.* 68P15, 03G10.

1998 *CR Categories and Descriptors.* H.2.1 [**D**atabase Management]: Logical design – Scheme and subschema.

Key words and phrases. XML data design, Formal Concept Analysis.

and it is expected that they play a similar role in XML databases, having specific properties and features due to the structure of XML data.

XML functional dependencies (XML FD) have become an important research topic. In 2004, Arenas and Libkin ([1]) adopted a tree tuple-based approach, defining for the first time an XML FD and a normal form. Yu and Jagadish prove in [19] that the previously introduced notions of XML FD are insufficient, and propose a generalized tree tuple-based XML FD.

Formal Concept Analysis offers an algebraic approach to data analysis and knowledge processing. Hence, it lies at hand to use FCA for mapping conceptual designs into XML schemas, since the XML data model is both hierarchical and semistructured. The notion of dependencies between attributes in a many-valued context has been introduced in [4], by Ganter and Wille. J. Hereth investigates in [6] how some basic concepts from database theory translate into the language of Formal Concept Analysis. He defines the power context family resulting from the canonical translation of a relational database. Regarding this power context family, he defines the formal context of functional dependencies. A detailed analysis and complex examples of the formal context of functional dependencies for a relational table are presented in [9]. Determining the implications in this context is investigated in [10], using a specially designed software. These are syntactically the same as functional dependencies in the analyzed table.

Uncovering functional dependencies in XML using FCA has been studied in [13]. AN XML document is read and the formal context corresponding to the flat representation of the XML data is constructed. XML data is converted into a fully unnested relation, a single relational table, and existing FD discovery algorithms are applied directly. The implications are exactly the functional dependencies in the analyzed XML data. This study is continued in [8] and [7]. Here a framework is proposed, which parses the XML document and constructs the formal context corresponding to the flat representation of the XML data. The concept lattice is a useful graphical representation of the analyzed XML document's elements and their hierarchy. Keys and functional dependencies in XML data are determined, as attribute implications in the constructed formal context. Then, the scheme of the XML document is transformed in GTT-XNF using the detected functional dependencies.

In this article the hierarchical representation of XML data from the paper of Yu and Yagadish (2008) is used to define a triadic FCA approach for a conceptual model of XML data. The novelty of the paper is this triadic approach and the proposed formal tricontext of functional dependencies with respect to a tuple class.

2. Mining Functional Dependency in XML Data

XML functional dependency has been defined in different ways, but no generally accepted definition exists. The main problem with defining functional dependency for XML databases is the absence of the definition of a tuple concept for XML. Arenas and Libkin defined tree tuples based upon Document Type Definition (DTD) schema [1]. In [13] an FCA based approach is given to find functional dependency in XML data as using the approach from [1]. In this approach, the XML document is read and then the formal context corresponding to the flat representation of the XML data is constructed. Here we derive the list of implications, these implications are exactly the functional dependencies in the analyzed flat representation of XML data.

Hartmann et al. [2, 3] define functional dependencies using the concept of tree homomorphism. Wang [16] compared different functional dependency definitions for XML and proposed a new definition of XML FD, which unifies and generalizes the surveyed XML FDs. All these XML FD definitions are based upon path expressions created from DTDs or XML Schema definitions.

Szabó and Benczúr [12] define the functional dependency concept on general regular languages, which is applicable to XML. They consider an XML document as a set of text fragments, each fragment being a string of symbols and the types of these strings are sentences of a regular language.

Yu and Jagadish [19] found that the tree tuples model of Arenas and Libkin [1] cannot handle set elements. They extend the tree tuple model as Generalized Tree Tuple (GTT) by incorporating set element type into the XML FD specification.

In [8] and its extended version [7], we propose a framework to mine FDs from an XML database; it is based on the notions of Generalized Tree Tuple, XML functional dependency and XML key notion as introduced by [19]. The formal context for a tuple class or the whole XML document is constructed from the flat representation of the generalized tree tuple. Non-leaf and leaf level elements (or attributes) and corresponding values are inserted in the formal context, then the concept lattice of the XML data is constructed. The obtained conceptual hierarchy is a useful graphical representation of the analyzed XML document's elements and their hierarchy. The software also finds the keys in the XML document. The set of implications resulted from this concept lattice will be equivalent to the set of functional dependencies that hold in the XML database. If the XML data representation is nested, solving the problem of mining XML FD's using FCA becomes more complicated and involves the use of multicontexts, tricontexts and power tricontexts families.

We start by recalling some basic definitions from [19].

Definition 1. (Schema) *A schema is defined as a set* $S = (E, T, r)$*, where:*

- *E is a finite set of element labels;*
- *T is a finite set of element types, and each $e \in E$ is associated with a $\tau \in T$, written as $(e : \tau)$, τ has the next form:* $\tau ::= \texttt{str} \mid \texttt{int} \mid \texttt{float} \mid \texttt{SetOf}\ \tau \mid \texttt{Rcd}[e_1 : \tau_1, \ldots, e_n : \tau_n]$*;*
- *$r \in E$ is the label of the root element, whose associated element type can not be* `SetOf` τ*.*

This definition contains some basic constructs in XML Scheme [15]. The types `str,int` and `float` are system defined *simple types* and `Rcd` indicate *complex scheme elements* (elements with children elements). Keyword `SetOf` is used to indicate *set schema elements* (elements that can have multiple matching data elements sharing the same parent in the data). We will treat attributes and elements in the same way, with a reserved "@" symbol before attributes.

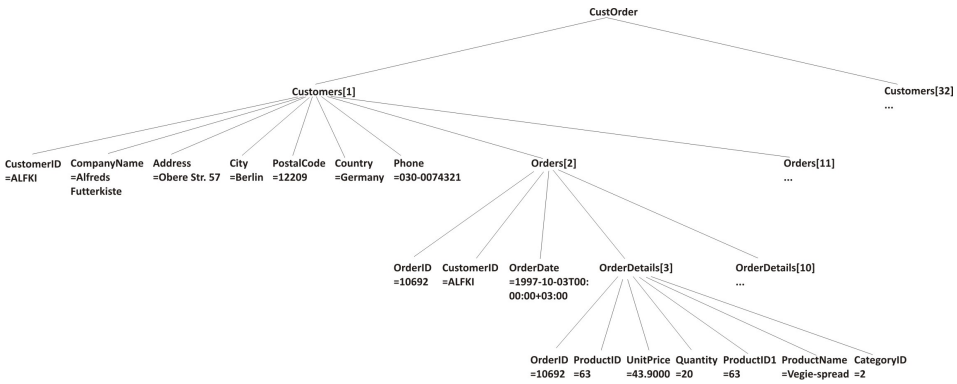


FIGURE 1. Example Tree

Example 1. The scheme $S_{CustOrder}$ of XML document from Figure 1 is:

```
CustOrder:Rcd
   Customers:SetOf Rcd
      CustomerID: str
      CompanyName: str
      Address: str
      City: str
      Country: str
      Phone: str
      Orders: SetOf Rcd
         OrderID: int
         CustomerID: str
```

```
OrderDate: str
OrderDetails: SetOf Rcd
    OrderID: int
    ProductID: int
    UnitPrice: float
    Quantity: float
    ProductName: str
    CategoryID: int
```

A schema element e_k can be identified through a path expression, $path(e_k) = /e_1/e_2/.../e_k$, where $e_1 = r$, and e_i is associated with type $\tau_i ::= \texttt{Rcd}\ [..., e_{i+1} : \tau_{i+1}, ...]$ for all $i \in [1, k-1]$. A path is *repeatable*, if e_k is a set element. We adopt XPath steps "." (self) and ".." (parent) to form a relative path given an anchor path.

Definition 2. (Data tree) *An XML database is defined to be a rooted labeled tree $T = \langle N, \mathcal{P}, \mathcal{V}, n_r \rangle$, where:*

- *N is a set of labeled data nodes, each $n \in N$ has a label e and a node key that uniquely identifies it in T;*
- *$n_r \in N$ is the root node;*
- *$\mathcal{P}$ is a set of parent-child edges, there is exactly one $p = (n', n)$ in $\mathcal{P}$ for each $n \in N$ (except n_r), where $n' \in N, n \neq n', n'$ is called the parent node, n is called the child node;*
- *$\mathcal{V}$ is a set of value assignments, there is exactly one $v = (n, s)$ in $\mathcal{V}$ for each leaf node $n \in N$, where s is a value of simple type.*

We assign a node key, referred to as @key, to each data node in the data tree in a pre-order traversal. A data element n_k is a descendant of another data element n_1 if there exists a series of data elements n_i, such that $(n_i, n_{i+1}) \in \mathcal{P}$ for all $i \in [1, k-1]$. Data element n_k can be addressed using a path expression, $path(n_k) = /e_1/\ldots/e_k$, where e_i is the label of n_i for each $i \in [1, k]$, $n_1 = n_r$, and $(n_i, n_{i+1}) \in \mathcal{P}$ for all $i \in [1, k-1]$.

A data element n_k is called *repeatable* if e_k corresponds to a set element in the schema. Element n_k is called a *direct descendant* of element n_a, if n_k is a descendant of n_a, $path(n_k) = \ldots/e_a/e_1/\ldots/e_{k-1}/e_k$, and e_i is not a set element for any $i \in [1, k-1]$.

In considering data redundancy, it is important to determine the equality between the "values" associated with two data elements, instead of comparing their "identities" which are represented by @key. So, we have:

Definition 3. (Element-value equality) *Two data elements n_1 of $T_1 = \langle N_1, \mathcal{P}_1, \mathcal{V}_1, n_{r1} \rangle$ and n_2 of $T_2 = \langle N_2, \mathcal{P}_2, \mathcal{V}_2, n_{r2} \rangle$ are element-value equal (written as $n_1 =_{ev} n_2$) if and only if:*

- *n_1 and n_2 both exist and have the same label;*
- *There exists a set M, such that for every pair $(n'_1, n'_2) \in M$, $n'_1 =_{ev} n'_2$, where n'_1, n'_2 are children elements of n_1, n_2, respectively. Every child element of n_1 or n_2 appears in exactly one pair in M.*
- *$(n_1, s) \in \mathcal{V}_1$ if and only if $(n_2, s) \in \mathcal{V}_2$,where s is a simple value.*

Definition 4. (Path-value equality) *Two data element paths p_1 on $T_1 = \langle N_1, \mathcal{P}_1, \mathcal{V}_1, n_{r1}\rangle$ and p_2 on $T_2 = \langle N_2, \mathcal{P}_2, \mathcal{V}_2, n_{r2}\rangle$ are path-value equal (written as $T_1.p_1 =_{pv} T_2.p_2$) if and only if there is a set M' of matching pairs where*

- *For each pair $m' = (n_1, n_2)$ in M', $n_1 \in N_1$, $n_2 \in N_2$, $path(n_1) = p_1$, $path(n_2) = p_2$, and $n_1 =_{ev} n_2$;*
- *All data elements with path p_1 in T_1 and path p_2 in T_2 participate in M', and each such data element participates in only one such pair.*

The definition of functional dependency in XML data needs the definition of so called Generalized Tree Tuple.

Definition 5. (Generalized tree tuple) *A generalized tree tuple of data tree $T = \langle N, \mathcal{P}, \mathcal{V}, n_r\rangle$, with regard to a particular data element n_p (called pivot node), is a tree $t^T_{n_p} = \langle N^t, \mathcal{P}^t, \mathcal{V}^t, n_r\rangle$, where:*

- *$N^t \subseteq N$ is the set of nodes, $n_p \in N^t$;*
- *$\mathcal{P}^t \subseteq \mathcal{P}$ is the set of parent-child edges;*
- *$\mathcal{V}^t \subseteq \mathcal{V}$ is the set of value assignments;*
- *n_r is the same root node in both $t^T_{n_p}$ and T ;*
- *$n \in N^t$ if and only if: 1) n is a descendant or ancestor of n_p in T, or 2) n is a non-repeatable direct descendant of an ancestor of n_p in T ;*
- *$(n_1, n_2) \in \mathcal{P}^t$ if and only if $n_1 \in N^t$, $n_2 \in N^t, (n_1, n_2) \in \mathcal{P}$;*
- *$(n, s) \in \mathcal{V}^t$ if and only if $n \in N^t$, $(n, s) \in \mathcal{V}$.*

A generalized tree tuple is a data tree projected from the original data tree. It has an extra parameter called a pivot node. In contrast with the notion of a tree tuple defined in [1], which separate sibling nodes with the same path at all hierarchy levels, the generalized tree tuple separate sibling nodes with the same path above the pivot node. An example of a generalized tree tuple is given in Figure 2. Based on the pivot node, generalized tree tuples can be categorized into tuple classes:

Definition 6. (Tuple class) *A tuple class C^T_p of the data tree T is the set of all generalized tree tuples t^T_n, where $path(n) = p$. Path p is called the pivot path.*

Definition 7. (XML FD) *An XML FD is a triple $\langle C_p, LHS, RHS\rangle$, (LHS for Left Hand Side part of FD and RH for Right Hand Side) written as $LHS \rightarrow$*

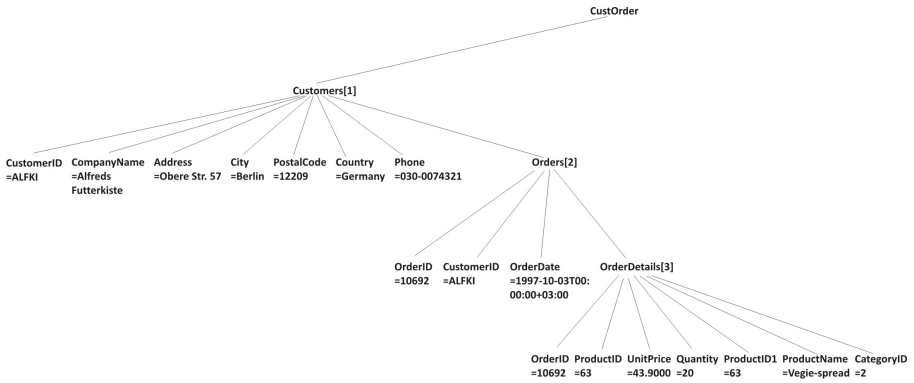


FIGURE 2. Example tree tuple

RHS w.r.t. C_p, where C_p denotes a tuple class, LHS is a set of paths (P_{li} , $i = [1, n]$) relative to p, and RHS is a single path (P_r) relative to p.

An XML FD holds on a data tree T (or T satisfies an XML FD) if and only if for any two generalized tree tuples $t_1, t_2 \in C_p$

- $\exists i \in [1, n]$, $t_1.P_{li} = \perp$ or $t_2.P_{li} = \perp$, or

- If $\forall i \in [1, n]$, $t_1.P_{li} =_{pv} t_2.P_{li}$, then $t_1.P_r \neq \perp, t_2.P_r \neq \perp, t_1.P_r =_{pv} t_2.P_r$.

A null value, $\perp$, results from a path that matches no node in the tuple, and $=_{pv}$ is the path-value equality defined in Definition 4.

Example 2. (XML FD) In our running example whenever two products agree on `ProductID` values, they have the same `ProductName`. This can be formulated as follows:
`{./ProductID}` $\rightarrow$ `./ProductName` w.r.t $C_{OrderDetails}$
Another example is:
`{./ProductID}` $\rightarrow$ `./CategoryID` w.r.t $C_{OrderDetails}$

In our approach we find the XML keys of a given XML document, so we need the next definition:

Definition 8. (XML key) *An XML Key of a data tree T is a pair $\langle C_p, LHS \rangle$, where T satisfies the XML FD $\langle C_p, LHS, ./@key \rangle$.*

Example 3. We have the XML FD: $\langle C_{Orders}, ./OrderID, ./@key \rangle$, which implies that $\langle C_{Orders}, ./OrderID \rangle$ is an XML key.

Tuple classes with repeatable pivot paths are called *essential tuple classes.*

Definition 9. (Interesting XML FD) *An XML FD $\langle C_p, LHS, RHS \rangle$ is interesting if it satisfies the following conditions:*

- *$RHS \notin LHS$;*

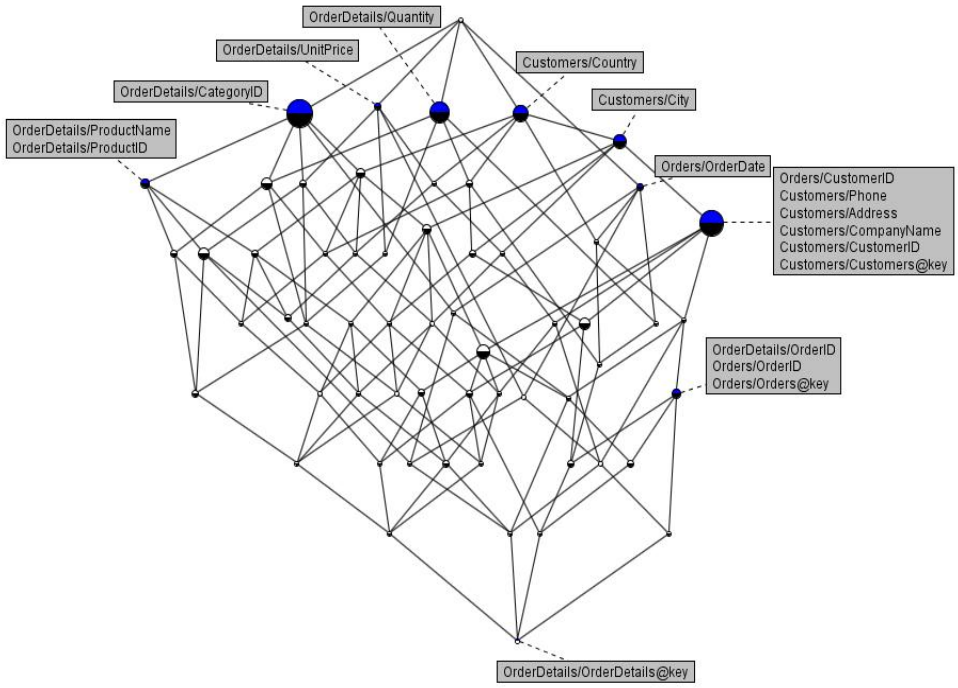


FIGURE 3. Concept Lattice of functional dependencies' Formal Context for tuple class $C_{Customers}$

- *C_p is an essential tuple class;*
- *RHS matches to descendent(s) of the pivot node.*

Definition 10. (XML data redundancy) *A data tree T contains a redundancy if and only if T satisfies an interesting XML FD $\langle C_p, LHS, RHS\rangle$, but does not satisfy the XML Key $\langle C_p, LHS\rangle$.*

As pointed out by [19], the hierarchical representation of XML data avoids many redundancies compared with the flat representation of XML data. Hence, we can separate individual relations like $R_{Customers}$, R_{Orders} or R_{Film}, R_{Actor} as in Tables 3 and 4. There are two types of relevant XML functional dependencies. The *intra-relational* XML FD's are the XML FD's whose LHS and RHS paths are in the same relation. Hence, an intrarelational FD is one that involves a single relation. As highlighted by [19], most of the interesting FD's in XML datasets are not intra-relational., i.e., they do not contain only LHS

or RHS paths within the same relation. Consider the XML data set in Figure 6. Then, intra-relational FD's are `SName` $\rightarrow$ `Founded`, `SName` $\rightarrow$ `film`, and `film/Title, film/Year` $\rightarrow$ `film/Director, film/actor` in R_{Studio}.

3. FCA Grounded Discovery of Intra-relational Functional Dependencies in XML Data

In order to mine intra-relational functional dependencies using FCA, we can use the same procedure as in the flat representation of XML datasets. This algorithm is presented in detail in [8] and [7], in the following we give just a sketch of how it is done.

Consider, the tuple class $C_{Customers}$. First, we construct the formal context of functional dependencies for XML data, see [7]. The concept lattice of this context is represented in Figure 3. The concept lattice displays also the hierarchy of the analyzed data. For instance, the node labeled $Customers/Country$ is on a higher level than the node labeled $Customers/City$. The $Customers$ node with six attributes is a subconcept of the concept labeled $Customers/City$. In our XML data, every customer has different name, address, phone number, so these attributes appear in one concept node and imply each other.

We can also observe, that the information about $Products$ is displayed on the other side of the lattice. $Products$ are in many-to-many relationships with $Orders$, linked by $OrderDetail$ in this case. The specially designed software FCAMineXFD mines the functional dependencies. A part of these XML FD-s are shown in Figure 4.

Given the set of dependencies discovered by this tool, we adopt the normalization algorithm of [19] to convert one XML schema into a correct one. The resulting scheme is shown in Figure 5.

4. Mining Inter-relational Functional Dependencies in XML Data

XML data, due to its specificity, has two different representations: a flat and hierarchical (non-flat) representation. XML elements may be simple elements but they also may nest other elements. Consider the XML document from Figure 6. It displays information extracted from a movie database concerning film studios, films, actors, etc.

The XML schema notation (XSN) allows to specify sequences and choices of elements. The scheme $S_{MoviesDB}$ of XML document in XSN from Figure 6 is given by:

```
MoviesDB (studio*)
   studio (SName, Founded, film*)
         film (Title, Year, Director, actor*)
```

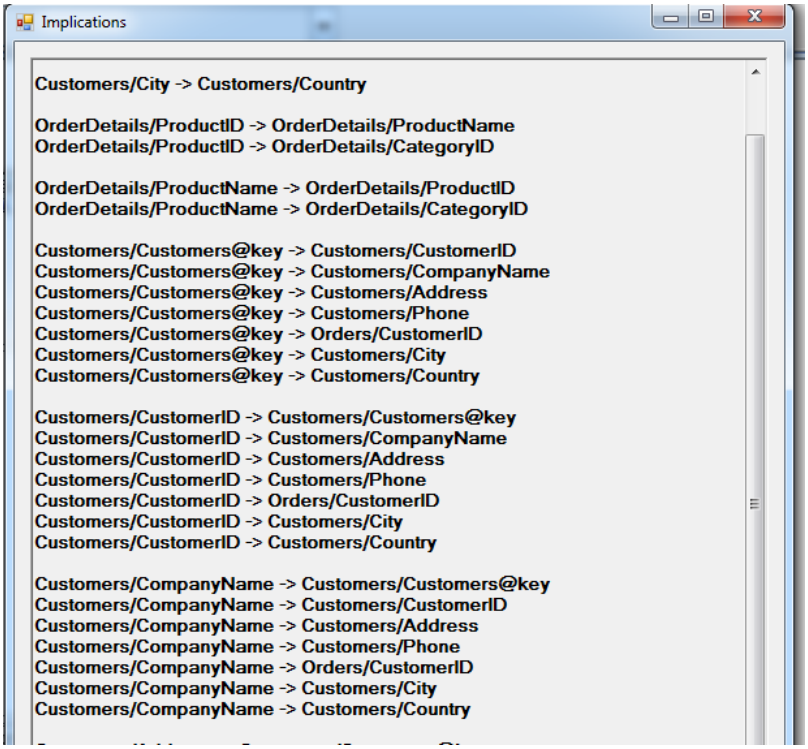


FIGURE 4. Functional dependencies in tuple class $C_{Customers}$

TABLE 1. Table R_{root}

@key	parent
1	$\perp$

TABLE 2. Table R_{Studio}

@key	parent	SName	Founded
10	1	Columbia Pictures	1924
50	1	Warner Bros. Pictures	1923

```
actor (AName, Gender, Born, BornY?)
```

In the flat representation, the data tree is represented as a single relational table. The hierarchical representation is more compact. The original XML tree is represented by a set of nested relations based on the XML schema, each relation R_p corresponds to an essential tuple class C_p (see [19]). In our movie

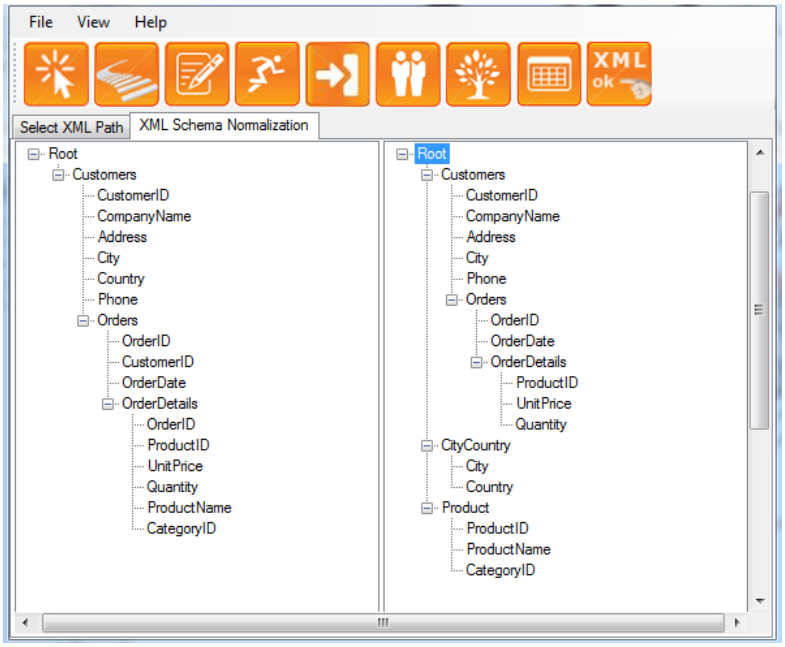


FIGURE 5. Correct XML Scheme

TABLE 3. Table R_{Film}

@key	parent	Title	Year	Director
13	10	Da Vinci Code	2006	Ron Howard
30	10	Captain Phillips	2013	Paul Greengrass
55	50	Extremely Loud & Incredibly Close	2011	Stephen Daldry
80	50	Gravity	2013	Alphonso Cuarón

TABLE 4. Table R_{Actor}

@key	parent	AName	Gender	Born	BornYear
20	13	Tom Hanks	M	USA	1956
25	13	Audrey Tautou	F	France	$\perp$
35	30	Tom Hanks	M	USA	1956
40	30	Barkhad Abdi	M	Somalia	$\perp$
60	55	Thomas Horn	M	USA	$\perp$
65	55	Tom Hanks	M	USA	1956
70	55	Sandra Bullock	F	USA	$\perp$
85	80	Sandra Bullock	F	USA	1964

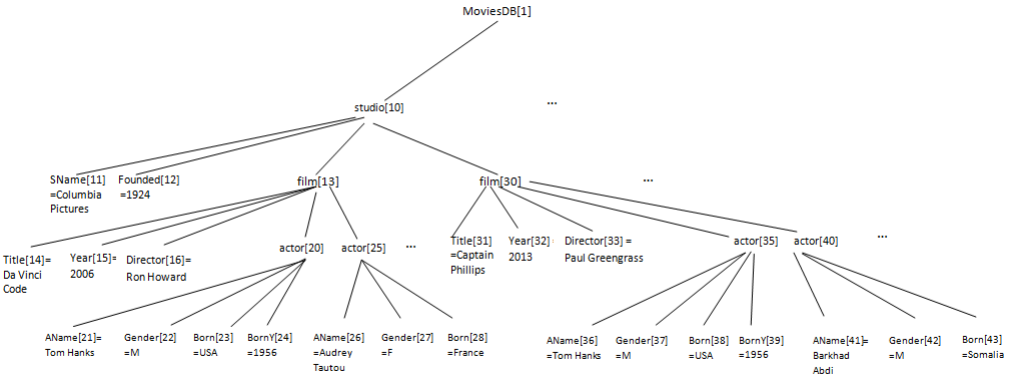


FIGURE 6. Movie Tree

dataset example, the hierarchical representation of the XML data from Figure 6 is given in the nested tables 1 - 4.

Every nested table is considered as a many-valued context. Through conceptual scaling, we obtain a multi-context wherefrom we can construct a tri-context.

In the following, we briefly recall some definitions.

Definition 11. A triadic formal context (shortly tricontext) is a quadruple $\mathbb{K} := (K_1, K_2, K_3, Y)$ where K_1, K_2 and K_3 are sets, and Y is a ternary relation between them, i. e., $Y \subseteq K_1 \times K_2 \times K_3$. The elements of K_1, K_2 and K_3 are called (formal) objects, attributes, and conditions, respectively. An element $(g, m, b) \in Y$ is read *object g has attribute m under condition b.*

Definition 12 ([17])**.** A multicontext of signature $\sigma\colon P \to I^2$, where I and P are non-empty sets, is be defined as a pair (S_I, R_P) consisting of a family $S_I := (S_i)_{i\in I}$ of sets and a family $R_P := (R_p)_{p\in P}$ of binary relations with $R_p \subseteq S_i \times S_j$ if $\sigma p = (i, j)$. A multicontext $\mathbb{K} := (S_I, R_P)$ can be understood as a *network* of formal contexts $\mathbb{K}_p := (S_i, S_j, R_p)$, with $p \in P$ and $\sigma p = (i, j)$. According to this understanding, the conceptual structure of a multicontext $\mathbb{K}$ is constituted by the concept lattices of its components $\mathbb{K}_p$.

In our example, every many-valued context representing a nested table of the XML dataset is nominally scaled. We obtain four contexts, which alltogether form the multicontext $\mathbb{K}_{Movie}$.

Example 4. The conceptual structure of the *Movie* multicontext is displayed in the Figures 7-10.

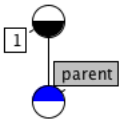


FIGURE 7. R_{root}

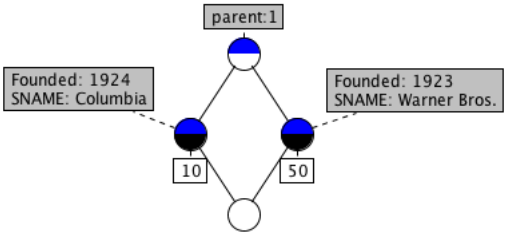


FIGURE 8. R_{studio}

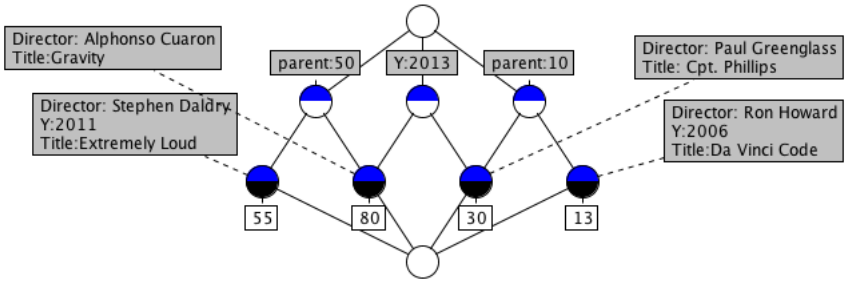


FIGURE 9. R_{film}

For a multicontext $\mathbb{K} := (S_I, R_P)$ of signature $\sigma\colon P \to I^2$, let $I_1 := \{i \in I \mid \sigma p = (i, j) \text{ for some } p \in P\}$ and $I_2 := \{j \in I \mid \sigma p = (i, j) \text{ for some } p \in P\}$.

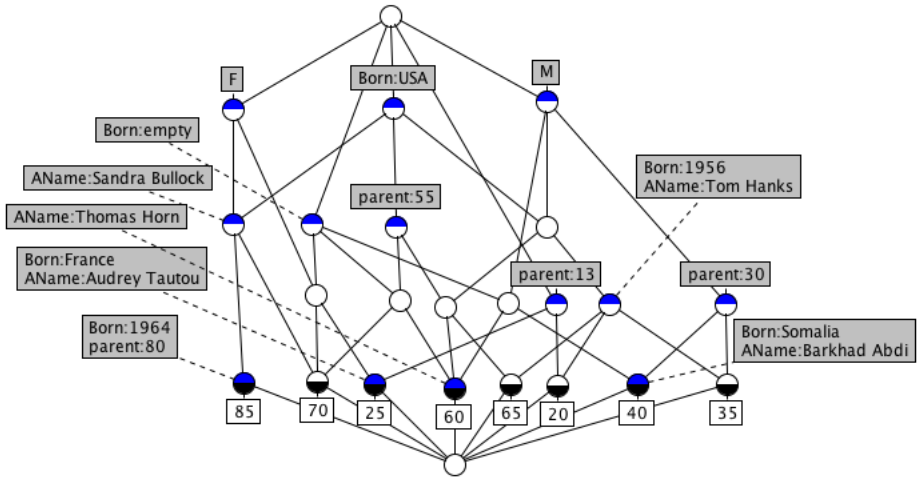


FIGURE 10. R_{actor}

Let $G_{\mathbb{K}} := \bigcup_{i \in I_1} S_i$ and $M_{\mathbb{K}} := \bigcup_{j \in I_2} S_j$. We can define a *triadic context* (for short tri-context) by $\mathbb{T}_{\mathbb{K}} := (G_{\mathbb{K}}, M_{\mathbb{K}}, P, Y_{\mathbb{K}})$ with $Y_{\mathbb{K}} := \{(g, m, p) \in G_{\mathbb{K}} \times M_{\mathbb{K}} \times P \mid (g, m) \in R_p\}$. The conceptual structure of $\mathbb{T}_{\mathbb{K}}$ can be seen as a natural triadic extension of the concept lattices $\mathfrak{B}(\mathbb{K}_p)$.

Definition 13. Given an XML database, the above construction gives us the *canonical translation* of the XML database as a formal tricontext.

Example 5. The triadic context generated by the *Movies* multicontext has as object set $G := \{1, 10, 50, 13, 30, 55, 80, 20, 25, 35, 40, 60, 65, 70, 85\}$, the attribute set is obtained by taking all nominally scaled attributes of the four many-valued contexts 1 - 4, and the condition set contains the labels of the four hierarchical levels from $\top$ to *Actor*.

Definition 14. For $\{i, j, k\} = \{1, 2, 3\}$ with $j < k$ and for $X \subseteq K_i$ and $Z \subseteq K_j \times K_k$, the $(-)^{(i)}$-derivation operators are defined by

$$X \mapsto X^{(i)} := \{(a_j, a_k) \in K_j \times K_k \mid (a_i, a_j, a_k) \in Y \text{ for all } a_i \in X\},$$

$$Z \mapsto Z^{(i)} := \{a_i \in K_i \mid (a_i, a_j, a_k) \in Y \text{ for all } (a_j, a_k) \in Z\}.$$

These derivation operators correspond to the derivation operators of the dyadic contexts defined by $\mathbb{K}^{(i)} := (K_i, K_j \times K_k, Y^{(i)})$, where

$$a_1 Y^{(1)}(a_2, a_3) \Leftrightarrow a_2 Y^{(2)}(a_1, a_3) \Leftrightarrow a_3 Y^{(3)}(a_1, a_3) \Leftrightarrow (a_1, a_2, a_3) \in Y.$$

Definition 15. For $\{i, j, k\} = \{1, 2, 3\}$ and for $X_i \subseteq K_i, X_j \subseteq K_j$ and $A_k \subseteq K_k$, the $(-)^{A_k}$-derivation operators are defined by

$$X_i \mapsto X_i^{A_k} := \{a_j \in K_j \mid (a_i, a_j, a_k) \in Y \text{ for all } (a_i, a_k) \in X_i \times A_k\},$$

$$X_j \mapsto X_j^{A_k} := \{a_i \in K_i \mid (a_i, a_j, a_k) \in Y \text{ for all } (a_j, a_k) \in X_j \times A_k\}.$$

These derivation operators correspond to the derivation operators of the dyadic contexts defined by $\mathbb{K}_{A_k}^{ij} := (K_i, K_j, Y_{A_k}^{ij})$ where

$$(a_i, a_j) \in Y_{A_k}^{ij} \Leftrightarrow (a_i, a_j, a_k) \in Y \text{ for all } a_k \in A_k.$$

Example 6. In the tricontext *Movies*, if g is an object, i.e., is a key of a node in the XML dataset about movies, then $i = 1, j = 2, k = 3$ and $g^{(1)}$ is the tree tuple having as root node the parent of g, while $g^{(K_3)}$ is the generalized tree tuple having g as a pivot element (see Definition 5).

This allows the algorithmic discovery of all generalized tree tuples with a given pivot element. These generalized tree tuples are playing an essential role in defining inter-relational functional dependencies as defined in [19]

If e is an element name, let A be the set of all nodes in the XML data set, having as label the element name e. Then $A^{(K_3)}$ is the tuple class of e (see Definition 6).

We have represented all important elements from an XML dataset using Triadic FCA. Discovering inter-relational functional dependencies can now be done using the algorithms developed for mining triadic implications.

Definition 16. ([5]) If $\mathbb{K} := (G, M, B, Y)$ is a tricontext, $R, S \subseteq M, C \subseteq B$, an expression of the form $R \overset{C}{\rightarrow} S$ is called *conditional attribute implication* and is read as R implies S under all conditions from C. A conditional attribute implication $R \overset{C}{\rightarrow} S$ *holds* in $\mathbb{K}$ if and only if the following is satisfied:

For each condition $c \in C$, it holds that if an object $g \in G$ has all the attributes in R then it also has all the attributes in S.

Definition 17. Let $\mathbb{K}$ be a the tricontext resulting from the canonical translation of an XML database. Let C_p be a tuple class. Then, the *formal tricontext of functional dependencies with respect to* C_p is defined as $\text{XMLFD}(\mathbb{K}) := (C_p \times C_p, M, P, Y)$, where M is the set of element names, P the set of nested tables, and $((g, h), e, p) \in Y$ if and only if the path values of g and h are equal with regard to path-value equality from Definition 4.

Proposition 1. The inter-relational functional dependencies of an XML database are exactly the conditional attribute implications in XMLFD($\mathbb{K}$).

5. Conclusion and Further Research

As far as described above, FCA proves to be a valuable tool for the conceptual design of XML data. XML data can be represented in hierarchical or flat form. In a recent work we give an FCA based approach for mining functional dependencies for flat XML data representation. In this paper we define a triadic FCA approach for a conceptual model of hierarchical XML data representation. The formal tricontext of functional dependencies with respect to a tuple class is given. This triadic approach is applicable in discovering inter-relational functional dependencies using algorithms developed for mining triadic implications.

As future work we propose to develop a software which will build the tricontext of an XML tree. The conditional attribute implications will give the functional dependencies from XML tree.

References

[1] M. Arenas, L. Libkin: *A normal form for XML documents.* TODS 29(1), pp. 195-232 (2004)

[2] S. Hartmann, S. Link: *More functional dependencies for XML.* In: Proc. ADBIS, pp. 355-369 (2003)

[3] S. Hartmann, S. Link, T. Trinh: *Solving the implication problem for XML functional dependencies with properties.* In: Logic, Language, Information and Computation, 17th International Workshop, WoLLIC, pp. 161-175 (2010)

[4] B. Ganter, R., Wille: *Formal Concept Analysis. Mathematical Foundations.* Springer, Berlin-Heidelberg-New York(1999)

[5] B. Ganter, S. Obiedkov, *Implications in Triadic Formal Contexts*, in ICCS 2004, LNAI 3127, pp. 186-195, Springer Verlag, 2004.

[6] J. Hereth: *Relational Scaling and Databases.* Proceedings of the 10th International Conference on Conceptual Structures: Integration and Interfaces LNCS 2393, Springer Verlag, pp. 62-76 (2002)

[7] K.T. Janosi-Rancz, V. Varga.: *XML Schema Refinement Through Formal Concept Analysis*, Studia Univ. Babes-Bolyai Cluj-Napoca, Informatica, vol. LVII, No. 3, pp. 49-64 (2012)

[8] K.T. Janosi-Rancz, V. Varga, T. Nagy: *Detecting XML Functional Dependencies through Formal Concept Analysis*, 14th East European Conference on Advances in Databases and Information Systems (ADBIS), Novi Sad, Serbia, LNCS 6295, pp. 595-598 (2010).

[9] K.T. Janosi-Rancz, V. Varga: *A Method for Mining Functional Dependecies in Relational Database Design Using FCA*, Studia Univ. Babeş-Bolyai, Informatica, Vol. LIII, Nr. 1 (2008), pp. 17-28.

[10] K.T. Janosi-Rancz, V. Varga, J. Puskas: *A Software Tool for Data Analysis Based on Formal Concept Analysis*, Studia Univ. Babeş-Bolyai, Informatica, Vol. LIII, Nr. 2 (2008), pp. 67-78.

[11] F. Lehmann, R. Wille: *A Triadic Approach to Formal Concept Analysis*, in: Ellis, G., Levinson, R., Rich, W., Sowa, J. F. (eds.), Conceptual Structures: Applications, Implementation and Theory, vol. 954 of Lecture Notes in Artificial Intelligence, Springer Verlag, (1995), pp. 32-43

[12] Gy. Szabó, A. Benczúr.: *Functional Dependencies on Extended Relations Defined by Regular Languages*, Annals of Mathematics and Artificial Intelligence, May 2013, pp. 1-39.

[13] V. Varga, K.T. Janosi-Rancz, C. Sacarea, K. Csioban: *XML Design: an FCA Point of View*, Proceedings of 2010 IEEE International Conference on Automation, Quality and Testing, Robotics, Theta 17th edition, Cluj Napoca, pp. 165-170 (2010)

[14] M. W. Vincent, J. Liu, C. Liu: *Strong functional dependencies and their application to normal forms in XML*, ACM TODS, 29(3), pp. 445-462 (2004)

[15] W3C. XML Schema, http://www.w3.org/XML/Schema (2014)

[16] J. Wang: *A comparative study of functional dependencies for XML*. In: APWeb, pp. 308-319 (2005)

[17] R. Wille: *Conceptual Structures of Multicontexts*. In Conceptual Structures: Knowledge Representation as Interlingua Lecture Notes in Computer Science Volume 1115, (1996), pp 23-39.

[18] S.A. Yevtushenko: *System of data analysis "Concept Explorer"*. (In Russian). Proceedings of the 7th national conference on Artificial Intelligence KII, Russia, pp. 127-134 (2000).

[19] C. Yu, H. V. Jagadish: *XML schema refinement through redundancy detection and normalization*, VLDB J. 17(2), pp. 203-223 (2008)

Babeş-Bolyai University, Cluj, Romania
E-mail address: csacarea@math.ubbcluj.ro

Babeş-Bolyai University, Cluj, Romania
E-mail address: ivarga@cs.ubbcluj.ro