Building a Domain Knowledge Model Based on a Concept Lattice Integrating Expert Constraints

My Thao Tang[1], Aleksey Buzmakov[2],
Yannick Toussaint[1], and Amedeo Napoli[1]

[1] LORIA (CNRS - Inria NGE - U. de Lorraine), Vandœuvre-lès-Nancy, France
[2] National Research University Higher School of Economics, Perm, Russia
my-thao.tang@loria.fr,avbuzmakov@hse.ru,
yannick.toussaint@loria.fr,amedeo.napoli@loria.fr

Abstract. Traditionally, FCA can be used as a tool for eliciting from data a class schema in the form of either a set of attribute implications or a concept lattice. However, such a schema does not necessarily fit the point of view of a domain expert for different reasons, e.g. noise, errors or exceptions in the data. For example, in the domain of animals, an expert may expect that the rule "mammal implies do not lay eggs" holds, while this may not be the case if the platypus is among the objects in the formal context. In this paper, we propose to bridge the possible gap between the representation model based on a concept lattice and the representation model of a domain expert. The knowledge of the expert is encoded as a set of attribute dependencies or constraints which is "aligned" with the set of implications provided by the concept lattice, leading to modifications in the original concept lattice. The method can be generalized for generating lattices satisfying constraints based on attribute dependencies and using extensional projections. This method also allows the experts to keep a trace of the changes occurring in the original lattice and the revised version, and to assess how concepts in practice are related to concepts automatically issued from data.

Keywords: Formal Concept Analysis, projection, attribute implication, attribute dependency

1 Introduction

Formal Concept Analysis (FCA) [1] is a classification method which is helpful for the conceptualisation step in building ontologies [2]. FCA elicits from data a class schema in the form of either a set of attributes implications or a concept lattice. The concept lattice can be interpreted as a knowledge model in the form of a concept hierarchy and the logical structures of formal concepts and concept lattices are effective in supporting human reasoning [3]. However, building knowledge bases is a cognitive process and does not obey to strict and formal rules as domain experts may understand the domain in a different way than what is represented in data. Thus, often there exists a gap between the representation

ISBN 978-5-600-01454-1, National Research University Higher School of Economics, 2016.

model based on a concept lattice and the representation model as imagined by a domain expert. In order to bridge this gap, researchers [4–6] have tried to integrate into lattices experts' knowledge in the form of dependencies between attributes. In these approaches, attribute dependencies serve as constraints that lead to more comprehensible structures of formal concepts for domain experts: formal concepts which satisfy the constraints are then provided to experts, and formal concepts which do not satisfy the constraints are disregarded.

Accordingly, in this paper, we introduce a formal method for integrating expert constraints into concept lattices in such a way that we can maintain the lattice structure as this structure is effective in supporting human reasoning [3]. Moreover, instead of providing only concepts that satisfy experts' knowledge, the method allows experts to keep a trace of changes occurring in the original lattice and the final constrained version, and to assess how concepts in practice are related to concepts automatically issued from data.

In this work, we "align" a set of given attribute dependencies with the set of implications provided by the concept lattice, leading to modifications in the original lattice. The method extends the definition of dependencies between single attributes introduced in [5] to the case of dependencies between attribute sets, and allows domain experts to have more possibilities for expressing constraints. We are able to build the constrained lattices without changing data and provide the trace of changes by using extensional projections [7, 8] over lattices. From an original lattice, two different projections produce two different constrained lattices, and thus, the disagreement between the representation model based on a concept lattice and the representation model as imagined by a domain expert is filled with projections.

The paper is organized as follows. Firstly, we introduce some basic notions that provide the foundations of our work. Next, we detail the method for generating constrained lattices, providing the trace of changes by using projections. Finally, we conclude our work and draw some perspectives over the approach.

2 Preliminaries

In the following, we introduce some important definitions that support our work. Firstly, we introduce attribute implications and dependencies and secondly, we describe projections in a concept lattice. For the following definitions, we use the notation of FCA in [1], where the formal context (G, M, I) is composed of a set of objects G, a set of attributes M and an incidence relation set $I \subseteq G \times M$. An example of such formal context is presented in Table 1.

2.1 Attribute Implication and Dependencies

Implications in a formal context represent dependency relations between attributes existing in data. In a nutshell, given an implication $x \rightarrow y$ we can understand that *"all objects having x also have y"*. Attribute implications can be read off directly from a formal context as stated in Definition 1.

Definition 1 (Attribute Implication [1]) *An implication between sets of attributes $X, Y \subseteq M$ in a formal context (G, M, I) is denoted by $X \rightarrow Y$, where every object having all the attributes from X has also all the attributes from Y, i.e. $X' \subseteq Y'$.*

Following Definition 1, attribute implications can be verified in the formal context and in the concept lattice thanks to Propositions 1 and 2.

Proposition 1 ([1]). *An implication $X \rightarrow Y$ between sets of attributes $X, Y \subseteq M$ holds in (G, M, I) iff $Y \subseteq X''$. It then automatically holds in the set of all concept intents in the concept lattice as well.*

Proposition 2 ([1]). *An implication $X \rightarrow Y$ between sets of attributes $X, Y \subseteq M$ holds in a concept lattice iff $X \rightarrow m$, $\forall m \in Y$, where $X \rightarrow m \iff (X', X'') \leq (m', m'')$. (m', m'') is the attribute concept of m.*

Animal	has_two_legs (m1)	lays_eggs (m2)	can_fly (m3)	has_wings (m4)	has_fins (m5)	has_feathers (m6)	has_milk (m7)	has_backbone (m8)	lives_in_ water (m9)
bear (g1)	x						x	x	
carp (g2)		x			x			x	x
chicken (g3)	x	x	x	x		x		x	
crab (g4)		x							x
dolphin (g5)			x		x		x	x	x
honeybee (g6)		x	x	x					
penguin (g7)	x	x		x		x		x	x
wallaby (g8)	x						x	x	

Table 1: Formal context of animals

For example, the implication `m7:has_milk` $\rightarrow$ `m8:has_backbone` holds in the formal context of Table 1. Different from implications, attribute dependencies do not arise from data. They are dependency relations that experts *expect* to exist as attribute implications. In the following, we provide the definition of attribute dependencies based on [5] and [6].

Definition 2 (Attribute Dependency [6]) *An attribute dependency is in the form $x \prec y$, where attribute x is less important than attribute y and the presence of x is not meaningful without the presence of y.*

Definition 3 (Formal Concept Satisfaction & Constrained Posets [5]) *A formal concept (A, B) satisfies an attribute dependency $x \prec y$ between attributes x and y iff whenever $x \in B$ then $y \in B$.*

(1) A concept lattice $\mathcal{L}$ constrained by an attribute dependency $x \prec y$, is the collection of all formal concepts from lattice $\mathcal{L}$ which satisfy $x \prec y$.
(2) A concept lattice $\mathcal{L}$ constrained by a set of attribute dependencies D, is the collection of all formal concepts from lattice $\mathcal{L}$ which satisfy all attribute dependencies in D.

Notice that both collections are partially ordered subsets of the original lattice $\mathcal{L}$ [5]. We will refer to them as *constrained posets of lattice $\mathcal{L}$ w.r.t. dependency $x \prec y$ (or w.r.t. the set of dependencies D)*. To illustrate Definitions 2 and 3 above, consider the formal context of animals in Table 1. The corresponding concept lattice is shown in Fig. 1. The constrained poset w.r.t. attribute dependency `can_fly` $\prec$ `has_wings` is the collection of formal concepts from the lattice circled in blue in Fig. 1.

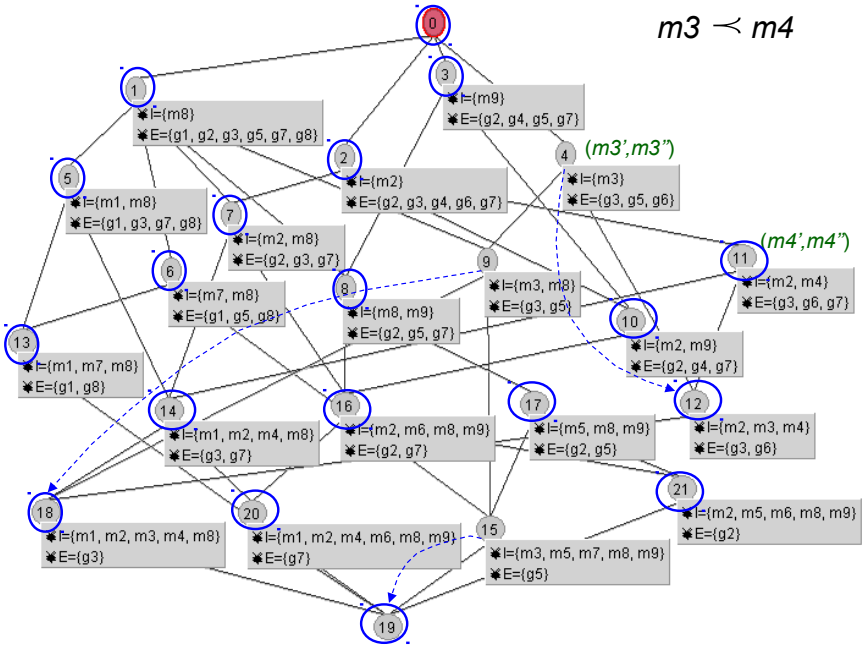

Fig. 1: The lattice constrained by `m3` $\prec$ `m4` and the trace of changes.

2.2 Projections

Projections are mathematical functions that allow mapping extents or intents in a concept lattice into "simpler" extents or intents, called *extensional* or *intensional* projections respectively [8, 9]. Initially, projections are used for simplifying descriptions of concepts in pattern structures [7, 10]. For our purpose, we use extensional projections.

Definition 4 (Extensional Projection [7]) *An extensional projection ψ is defined as a mapping verifying the following properties for each pair (A_1, A_2) of extents of a lattice $\mathcal{L}$: (1) if $A_1 \subseteq A_2$, then $\psi(A_1) \subseteq \psi(A_2)$ (monotone), (2) $\psi(A_1) \subseteq A_1$ (contractive), and (3) $\psi(\psi(A_1)) = \psi(A_1)$ (idempotent).*

Let $\mathcal{L}$ be a lattice, and ψ be an extensional projection of $\mathcal{L}$, then the set of extents E in $\mathcal{L}$ can be divided into two sets: $E = \{e \in E | \psi(e) = e\} \cup \{e \in E | \psi(e) \subset e\}$. The set $\{e \in E | \psi(e) = e\}$ is called the *fixed point* of ψ.

The mapping of a concept in a lattice onto the corresponding concept in the projected lattice can be computed thanks to Proposition 3.

Proposition 3 ([9]). *Let $\mathcal{L}$ be a lattice, and ψ be an extensional projection of $\mathcal{L}$, then a concept (A, B) in $\mathcal{L}$ is projected in the corresponding lattice $\psi(\mathcal{L})$ onto the concept (A_1, B_1) such that $A_1 = \psi(A)$ and $B_1 = A_1'$.*

It is worth noticing that the "projected lattice" which is the set of all projected concepts (or extents) is actually a concept lattice. For our purposes, this adds the benefit that the result of constraining a lattice through a projection preserves the lattice structure. Hereafter, we will refer to the result of constraining the lattice through a projection as a *constrained lattice*.

3 Projections for Generating Constrained Lattices

As previously discussed, in a given formal context, there exists some implications that represent attribute dependencies among attributes. However, implications which can be checked within the concept lattice are usually not in the data. For example, some objects may have missing attribute associations or may have wrongly assigned attributes. In a different scenario, an expert may simply want to observe formal concepts aligned through her particular point-of-view of the domain. Thus, often there exists a *disagreement* between the relations of attributes in the data and the relations of attributes as a domain expert understands them.

3.1 Discussion about Constrained Lattices

In order to bridge the disagreement between the representation model based on a concept lattice and the representation model as imagined by a domain expert, we "align" a set of given attribute dependencies with the set of implications provided by the concept lattice. According to Definitions 1 and 2, if an implication $x \rightarrow y$

holds in a lattice, then that lattice satisfies the attribute dependency $x \prec y$. To build a lattice satisfying a set of attribute dependencies, we look for a lattice that satisfies the corresponding set of attribute implications. In our setting, we want to use a well-founded process based on projections. Thus, we provide a method for projecting the lattice in such a way that the lattice structure is preserved. Moreover, we provide experts with a mapping of concepts in the original lattice onto the corresponding concepts in the revised version to make visible the changes in the lattice. We refer to these mappings as the *trace of changes* occurring in the original lattice and the revised version. We achieve this by using extensional projections over lattices.

We illustrate this scenario where a lattice $\mathcal{L}$ is mapped onto a lattice $\mathcal{L}_1$ which is a revised version w.r.t. the attribute dependency $x \prec y$. Let us call this mapping ς.

We observe the following characteristics of ς: (i) ς reduces the size of the lattice $\mathcal{L}$ because the constrained lattice $\mathcal{L}_1$ do not contain formal concepts in $\mathcal{L}$ which do not satisfy the constraints; (ii) In order to get formal concepts in $\mathcal{L}$ satisfying the constraints, ς replaces the concept extents in that lattice with smaller sets of objects which are still extents. This replacement may result in a loss of information. Hence, the trace of changes should be kept as it may be useful for domain experts to be aware that some concepts in the lattice will be lost some important objects. Indeed, (i) and (ii) are consequences of the fact that ς is an extensional projection. ς is a special case of projections from [7, 8]. This extensional projection does not create new extents, it replaces the concept extents in the lattice with smaller extents.

In the following, we describe how extensional projection is defined in two different cases, namely for a single attribute dependency and for a set of attribute dependencies.

3.2 Projections for Constrained Lattices w.r.t. Dependencies between Attribute Sets

Let us consider the problem of finding an extensional projection $\psi : \mathcal{L} \to \mathcal{L}_1$, where $\mathcal{L}$ is a concept lattice which does not satisfy the implication $x \to y$ between attributes $x, y \in M$, $\mathcal{L}_1$ is the projected lattice of $\mathcal{L}$ which satisfies the implication $x \to y$.

The following propositions state the main properties of ψ.

Proposition 4. *Let $\mathcal{L}$ be a concept lattice which does not satisfy the implication $x \to y$, then $x' \not\subseteq y' \Longrightarrow x' \cap y' \subset x'$.*

Proposition 5. *Let $\mathcal{L}$ be a concept lattice which does not satisfy the implication $x \to y$, and ψ be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies $x \to y$, then $\psi(x') = x' \cap y'$.*

Proof. see Appendix

Proposition 5 gives us an important property of ψ to observe changes in lattice $\mathcal{L}$: From lattice $\mathcal{L}$ which does not satisfy the implication $x \to y$, we are going to project x' in lattice $\mathcal{L}$ to $x' \cap y'$ to get the projected lattice that satisfies the implication $x \to y$. And this follows the properties of projections as $\psi(x') = x' \cap y' \subset x'$.

Given an extensional projection ψ of $\mathcal{L}$ such that $\psi(x') = x' \cap y'$, extents A in $\mathcal{L}$ can be divided into three categories as shown in Fig. 2.

- Category I contains all extents A that are subsumed by $x' \cap y'$, i.e. $A \subseteq (x' \cap y') \subset x'$ ($x' \cap y' \subset x'$ by Proposition 4).
- Category II contains all extents A that are subsumed by x' but not subsumed by $x' \cap y'$, i.e. $A \subseteq x'$, $A \not\subseteq (x' \cap y')$.
- Category III contains extents A that are not in categories I, II, i.e. $A \not\subseteq x'$.

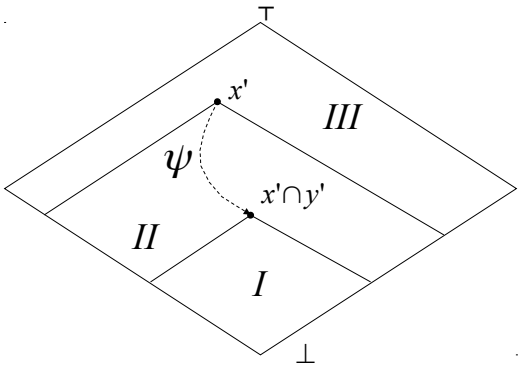

Fig. 2: Three possible categories of extents in lattice $\mathcal{L}$ when $\psi(x') = x' \cap y'$.

Consider an element A in Category I. As the objects in the set $x' \cap y'$ have both attributes x and y, the concept with extent $x' \cap y'$ in $\mathcal{L}$ satisfies the implication $x \to y$. Because A is subsumed by $x' \cap y'$, the concept with extent A in $\mathcal{L}$ satisfies $x \to y$ and remains the same in the projected lattice, i.e. $\psi(A) = A$. Category I is a component of the fixed point of the projection ψ.

Consider an element A in Category III. Because the objects in A do not have attribute x, the concept with extent A in $\mathcal{L}$ satisfies the implication $x \to y$ and remains the same in the projected lattice, i.e. $\psi(A) = A$. Category III is also a component of the fixed point of the projection ψ.

Consider an element A in Category II. We have: 1) $\psi(A) \subseteq A$ by the contractive property of projections; 2) As $A \subseteq x'$, $\psi(A) \subseteq \psi(x')$ by the monotonic property of projections. Moreover, $\psi(x') = x' \cap y'$ by Proposition 5. So, $\psi(A) \subseteq x' \cap y'$. As a result of 1) and 2), $\psi(A) \subseteq A \cap (x' \cap y')$.

In order to have the largest fixed point, we set $\psi(A) = A \cap (x' \cap y')$. $\psi(A) = A \cap (x' \cap y')$ complies with the properties of projections and the concept with extent $\psi(A)$ satisfies $x \to y$ because objects in $\psi(A) = A \cap (x' \cap y')$ have both attributes x and y. This introduces the fact that Category II constrains concepts which are projected in such a way that $\psi(A) = A \cap (x' \cap y')$.

Thus, the projection with the largest fixed point given by $\psi(x') = x' \cap y'$ is:

$$\psi(A) = \begin{cases} A \cap (x' \cap y') & \text{if } A \subseteq x', A \not\subseteq (x' \cap y'), \\ A & \text{otherwise.} \end{cases} \quad (1)$$

This projection gives the projected lattice satisfying the implication $x \to y$.

This projection can be extended to support dependencies between attribute sets of the form $X \prec Y$, where $X, Y \subseteq M$. In such a case, the projection is defined as:

$$\psi(A) = \begin{cases} A \cap (X' \cap Y') & \text{if } A \subseteq X', A \not\subseteq (X' \cap Y'), \\ A & \text{otherwise.} \end{cases} \quad (2)$$

The trace of changes occurring in the original lattice $\mathcal{L}$ and the constrained lattice $\mathcal{L}_1$ can be obtained thanks to Proposition 3.

Example 1. Consider the running example where the expert provides her knowledge in the form of an attribute dependency `m3:can_fly` $\prec$ `m4:has_wings`. According to the data in Table 1: `m3`$'$={`g3, g5, g6`}, `m4`$'$={`g3, g6, g7`}, `m3`$' \cap$ `m4`$'$={`g3, g6`}.

Applying Equation 2, the extensional projection ψ for generating the constrained lattice $\mathcal{L}_1$ from the original lattice $\mathcal{L}$ is:

$$\psi(A) = \begin{cases} A \cap \{\texttt{g3, g6}\} & \text{if } A \subseteq \{\texttt{g3, g5, g6}\}, A \not\subseteq \{\texttt{g3, g6}\}, \\ A & \text{otherwise.} \end{cases} \quad (3)$$

Fig. 1 depicts the constrained lattice and the trace of changes provided by the projection ψ. In Fig. 1, the formal concepts of the constrained lattice are circled blue. The transformations correspond to: $C_4 \to C_{12}$ ({`g3, g5, g6`} $\to$ {`g3, g6`}), $C_9 \to C_{18}$ ({`g3, g5`} $\to$ `g3`), and $C_{15} \to C_{19}$ ($g5 \to \emptyset$). The other transformation {`g5, g6`} $\to g6$ does not apply as neither {`g5, g6`} or $g6$ are extents in lattice $\mathcal{L}$.

An interpretation of the change C_4 in $\mathcal{L}$ to C_{12} in $\mathcal{L}_1$ according to the semantics of the extensional projection is as follows. According to the data, objects `g3:chicken`, `g6:honeybee` are grouped together with object `g5:dolphin` to form concept C_4 whose intent is {`m3:can_fly`}. According to the expert, animals that can fly should also have wings (`can_fly` $\prec$ `has_wings`), `g5:dolphin` should not be grouped together with `g3:chicken` and `g6:honeybee` to form a concept. This is better represented by concept C_{12} whose extent is {`g3:chicken`, `g6:honeybee`} and intent is {`m2:lays_eggs`, `m3:can_fly`, `m4:has_wings`}. By checking this change, we found that the data contain a noisy element: dolphins can fly.

3.3 Projections for Constrained Lattices w.r.t. Sets of Dependencies

A "naive" way to generate a constrained lattice satisfying a set of dependencies consists in generating first a constrained lattice for each dependency, and then getting the final constrained lattice by the intersection of these constrained lattices. A question raised here: is there a "good" way to generate the constrained lattice? In the following, we will show that dependencies should be treated following an order of projections.

Definition 5 ([9, 10]) *Let $\mathcal{L}$ be a concept lattice, and ψ_1, ψ_2 be two extensional projections of $\mathcal{L}$, we say that $\psi_1 \leq \psi_2$, iff there is some projection ψ defined on $\psi_2(\mathcal{L})$ such that for all extent A in $\mathcal{L}$, $\psi_1(A) = \psi \circ \psi_2(A)$. If $\psi_1(\mathcal{L}) \subseteq \psi_2(\mathcal{L})$, then $\psi_1 \leq \psi_2$.*

Definition 5 states that actually projections can be ordered from "general" to "specific". If ψ_1 is a projection over the projected lattice of ψ_2, then we say ψ_1 is more *specific* than ψ_2 or ψ_2 is more *general* than ψ_1.

There is a partial order on projections of a lattice given by Proposition 6. This proposition has been proven in [10].

Proposition 6 ([10]). *Extensional projections of a lattice $\mathcal{L}$ ordered by Definition 5 form a semi-lattice $(\mathcal{F}, \wedge)$, where the semi-lattice operation between $\psi_1, \psi_2 \in \mathcal{F}$ is given by $\psi_1 \wedge \psi_2 = \psi_3$ iff $\psi_3(\mathcal{L}) = \psi_1(\mathcal{L}) \cap \psi_2(\mathcal{L})$.*

The order on projections for generating constrained lattices can be characterized by Proposition 7.

Proposition 7. *Let $\mathcal{L}$ be a concept lattice, ψ_1 be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies the implication $X_1 \rightarrow Y_1$, and ψ_2 be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies the implication $X_2 \rightarrow Y_2$, where $X_1, Y_1, X_2, Y_2 \subseteq M$, we have:*

1) *If $X_1 \subseteq X_2$ and $Y_1 \subseteq Y_2$, then $\psi_1 \leq \psi_2$.*
2) *If $\psi_1 \leq \psi_2$, then the projected lattice given by ψ_1 satisfies $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$.*
3) *There exists an extensional projection ψ_3 of $\mathcal{L}$ such that $\psi_3(\mathcal{L}) = \psi_1(\mathcal{L}) \cap \psi_2(\mathcal{L})$. ψ_3 gives the constrained lattice that satisfies $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$.*

As a result of Proposition 7, given two dependencies between attribute sets, we can order the projections corresponding to these dependencies as follows. First, if one dependency depends on attribute sets that are included in the attribute sets of the other dependency, then the projection of that dependency is more specific than the projection of the other. Second, if one projection is more specific than the other, then we can use the more specific projection for generating the final constrained lattice . Third, we can use the projection that is the meet of the two projections for generating the final constrained lattice.

Let us now go back to our scenario of generating a constrained lattice that satisfies a set of dependencies. Let $\mathcal{L}$ be a concept lattice, and ψ_i be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies an implication

$X_i \to Y_i$, where $X_i, Y_i \subseteq M$. The set of projections ψ_i can be ordered according to the order of projections given by Proposition 7. This order forms a semi-lattice given by Proposition 6. By Proposition 7, the projection that is the meet of the most specific projections in this order gives the final constrained lattice satisfying the set of implications.

According to the order of projections, we have two possible ways of generating constrained lattices. The first way uses the meet of the most specific projections in the order of the set of projections to generate the final constrained lattice. The second way uses all the projections to generate a set of constrained lattices. Applying the first or the second way to generate constrained lattices depends on what experts need. The first way using the meet of the most specific projections to generate the final constrained lattice is more efficient in computation than the second way using all the projections to generate the corresponding constrained lattices. However, by using the meet of the most specific projections to generate the final constrained lattice, the first way only provides the trace of changes between the original lattice and the final constrained lattice. In the case experts need all the traces of changes, we need the second way using all the projections to generate the corresponding constrained lattices.

Example 2. Consider the running example where the expert provides her knowledge in the form of a set of dependencies d_i:

d_1) $\{\texttt{m3, m8}\} \prec \{\texttt{m4}\}$,
d_2) $\texttt{m3} \prec \texttt{m4}$,
d_3) $\{\texttt{m1, m2}\} \prec \{\texttt{m3}\}$,
d_4) $\texttt{m4} \prec \texttt{m3}$.

Let ψ_i be the extensional projection for generating the constrained lattice satisfying dependency d_i, applying Equation 2, we have:

$$\texttt{For } d_1 : \psi_1(A) = \begin{cases} A \cap \{\texttt{g3}\} & \text{if } A \subseteq \{\texttt{g3, g5}\}, A \not\subseteq \{\texttt{g3}\}, \\ A & \text{otherwise.} \end{cases} \tag{4}$$

$$\texttt{For } d_2 : \psi_2(A) = \begin{cases} A \cap \{\texttt{g3, g6}\} & \text{if } A \subseteq \{\texttt{g3, g5, g6}\}, A \not\subseteq \{\texttt{g3, g6}\}, \\ A & \text{otherwise.} \end{cases} \tag{5}$$

$$\texttt{For } d_3 : \psi_3(A) = \begin{cases} A \cap \{\texttt{g3}\} & \text{if } A \subseteq \{\texttt{g3, g7}\}, A \not\subseteq \{\texttt{g3}\}, \\ A & \text{otherwise.} \end{cases} \tag{6}$$

$$\texttt{For } d_4 : \psi_4(A) = \begin{cases} A \cap \{\texttt{g3, g6}\} & \text{if } A \subseteq \{\texttt{g3, g6, g7}\}, A \not\subseteq \{\texttt{g3, g6}\}, \\ A & \text{otherwise.} \end{cases} \tag{7}$$

Lattices in Figs. 3, 4, 5 and 6 depict the constrained lattices and the traces of changes provided by the projections ψ_1, ψ_2, ψ_3, ψ_4 respectively. In this set

of projections, $\psi_2 \leq \psi_1$ and $\psi_4 \leq \psi_3$. We can see that the constrained lattice $\psi_2(\mathcal{L})$ is included in $\psi_1(\mathcal{L})$ and $\psi_4(\mathcal{L})$ is included in $\psi_3(\mathcal{L})$.

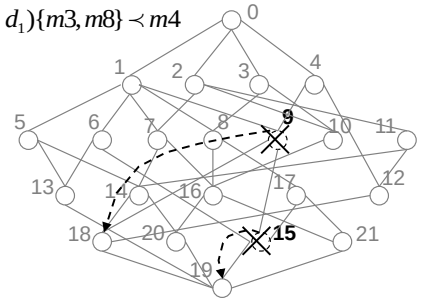

Fig. 3: The constrained lattice for d_1.

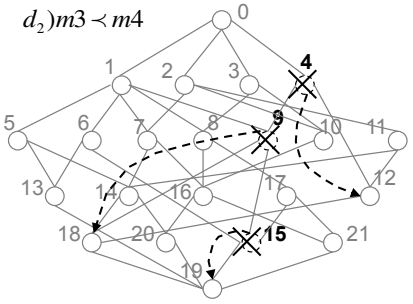

Fig. 4: The constrained lattice for d_2.

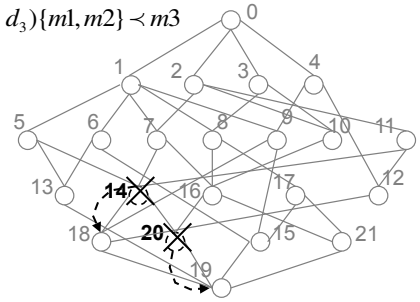

Fig. 5: The constrained lattice for d_3.

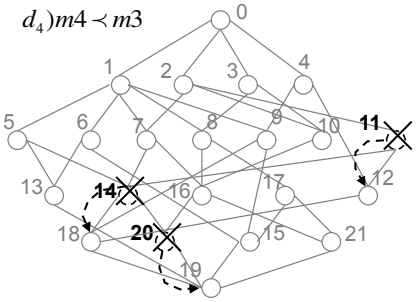

Fig. 6: The constrained lattice for d_4.

Let ψ_5 be an extensional projection that is the meet of the most specific projections: $\psi_5 = \psi_2 \wedge \psi_4$. ψ_5 can be defined such that:

$$\psi_5(A) = \begin{cases} A \cap \{\texttt{g3, g6}\} & \text{if } A \subseteq \{\texttt{g3, g6, g7}\}, A \not\subseteq \{\texttt{g3, g6}\}, \\ A \cap \{\texttt{g3, g6}\} & \text{if } A \subseteq \{\texttt{g3, g5, g6}\}, A \not\subseteq \{\texttt{g3, g6}\}, \\ A & \text{otherwise.} \end{cases} \quad (8)$$

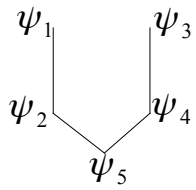

Fig. 7: The semi-lattice of projections.

The set of projections ψ_i forms a semi-lattice as shown in Fig. 7. We have two ways of generating the final constrained lattice. The first way uses the meet ψ_5 of the most specific projections. Lattice in Fig. 8 depicts the final constrained lattice and the trace of changes between the original lattice and the final constrained lattice provided by the projection ψ_5. The second way uses all the projections to have all the traces of changes. According to the semi-lattice of the projections, because $\psi_2 \leq \psi_1$ and $\psi_4 \leq \psi_3$, in order to get all the traces of changes, ψ_1 is applied before ψ_2 and ψ_3 is applied before ψ_4. By contrast, as ψ_2 and ψ_4 are incompatible, it does not matter if ψ_1 and ψ_2 or ψ_3 and ψ_4 are applied first. Thus, the projections can be applied according to either the order ψ_1,ψ_2,ψ_3,ψ_4,ψ_5 or ψ_3,ψ_4.ψ_1,ψ_2,ψ_5. The trace of changes can be either the chain of lattices in Figs. 3, 4, 5, 6, 8 or lattices in Figs. 5, 6, 3, 4, 8. In both cases, experts still can access the changes corresponding to each dependency.

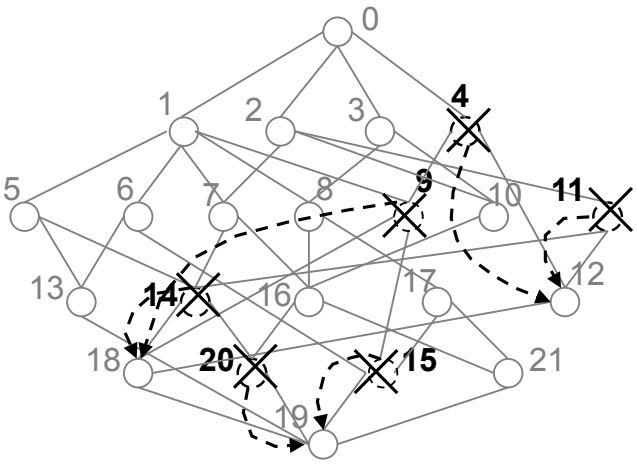

Fig. 8: The final constrained lattice and the trace of changes.

4 Related Work, Discussion and Conclusion

Taking into account expert knowledge in the form of dependencies between attributes in concept lattices has been proposed by several researchers [11, 12, 4, 5]. [11, 12] extended attribute exploration to include background implications, i.e. the implications that experts already know to be valid. In these approaches, they trust the original data and experts have to provide new objects as counterexamples. To deal with a situation such that experts know dependencies between attributes, but they do not know any new objects to provide, the other approaches

[4, 5] were proposed to build a concept hierarchy from a formal context extracted from data and from a hierarchy of attributes provided by domain experts. Two possible ways of adding knowledge to object descriptions were discussed in [4]. One way is to expand the formal context by adding to the description of each object all attributes that are implied by the original attributes. Expanding the context can lead to lose the original data and the formal context may become very large. The other way is to work with an unexpanded formal context by adapting the construction algorithms of lattices to extract formal concepts satisfying dependencies. A similar idea was proposed by [5], in which the authors adapted an incremental algorithm for computing constrained posets. By contrast, we do not expand the formal context nor adapt the construction algorithms of lattices. Our method uses projections to generate constrained lattices instead of constrained posets and to provide the trace of changes.

To conclude, in this paper, we have presented a formal method based on extensional projections for integrating expert knowledge into concept lattices in such a way that the lattice structure and the trace of changes are preserved. The expert knowledge is encoded as a set of attribute dependencies which is aligned with the set of implications provided by the concept lattice. According to the order of projections, the method offers two ways of generating constrained lattices. The first way uses the meet of the most specific projections to generate the final constrained lattice. The second way uses all the projections to generate a set of constrained lattices. The first way is more efficient in computation, but provides only the trace of changes between the original lattice and the final constrained lattice while the second way provides all the traces of changes, but less efficient in computation.

Currently, we are implementing the method for experiments with large datasets. Future work includes defining intensional projections to integrate dependencies between object sets. This is useful for many applications, e.g. when classifying documents, this can be applied to integrate a partial order of documents from experts into concept lattices. Another interesting application could be to complete definitions in data, e.g. the method presented in this paper can be applied to add implications that experts expect to exist.

Appendix

Proof (Proposition 5).

1) As the projected lattice satisfies the implication $x \to y$, $(\psi(y'), \psi(y')') \geq (\psi(x'), \psi(x')')$ (by Proposition 2). So, $\psi(x') \subseteq \psi(y')$.
2) $\psi(y') \subseteq y'$ (by the contractive property of projections).
3) As a result of 1) and 2), we have $\psi(x') \subseteq \psi(y') \subseteq y'$.
4) $\psi(x') \subseteq x'$ (by the contractive property of projections).
5) As a result of 3) and 4), we have $\psi(x') \subseteq x' \cap y'$.
6) For any $m \in M$, m' is the maximal set of objects having m in $\mathcal{L}$. Since ψ is contractive, this is also true in the projected lattice. Thus, $\psi(x')$ is the maximal set of objects having attribute x in the projected lattice $\mathcal{L}_1$.

7) Because the objects in the set $x' \cap y'$ have both attributes x and y, the concept with extent $x' \cap y'$ in $\mathcal{L}$ satisfies $x \to y$ and remains the same in $\mathcal{L}_1$. So, $x' \cap y'$ exists in $\mathcal{L}_1$ and the objects in this set have attribute x.
8) As a result of 6) and 7), we have $x' \cap y' \subseteq \psi(x')$.

As a result of 5) and 8), $\psi(x') = x' \cap y'$.

Proof (Proposition 7).

1) Let $\psi_1(\mathcal{L}), \psi_2(\mathcal{L})$ be the projected lattices given by ψ_1, ψ_2 respectively. As $\psi_1 \leq \psi_2$, according to Definition 5, $\exists\psi$: $\psi_1(\mathcal{L}) = \psi \circ \psi_2(\mathcal{L})$. Hence, $\psi_1(\mathcal{L}) \subseteq \psi_2(\mathcal{L})$. So, the projected lattice given by ψ_1 satisfies $X_1 \to Y_1$ and $X_2 \to Y_2$.
2) In order to give proof that $\psi_1 \leq \psi_2$, we will show that a projection ψ can be defined on $\psi_2(\mathcal{L})$ such that $\psi_1(\mathcal{L}) = \psi \circ \psi_2(\mathcal{L})$:

$$\psi_1(A) = \begin{cases} \psi_2(A) \cap (X_1' \cap X_2') & \text{if } A \not\subseteq X_2', A \not\subseteq X_1' \cap Y_1', A \subseteq X_1', \\ \psi_2(A) & \text{otherwise.} \end{cases} \quad (9)$$

3) This follows from Proposition 6 that the set of projections $\mathcal{F}$ over $\mathcal{L}$ is a semi-lattice, then the meet $\psi_3 = \psi_1 \wedge \psi_2$ must exist.

References

1. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer (1999)
2. Poelmans, J., Ignatov, D., Kuznetsov, S., Dedene, G.: Formal concept analysis in knowledge processing: A survey on applications. Expert Syst. Appl. (2013) 6538–6560
3. Wille, R.: Why can concept lattices support knowledge discovery in databases? Exp. Theor. Artif. Intell. (2002) 81–92
4. Carpineto, C., Romano, G.: Adding knowledge to concept lattices. In: Concept Data Analysis: Theory and Applications. John Wiley & Sons (2004) 64–81
5. Belohlavek, R., Sklenar, V.: Formal concept analysis constrained by attribute-dependency formulas. In: Formal Concept Analysis. Springer (2005) 176–191
6. Messai, N., Devignes, M.D., Napoli, A., Smail-Tabbone, M.: Extending attribute dependencies for lattice-based querying and navigation. In: ICCS. (2008) 189–202
7. Ganter, B., Kuznetsov, S.: Pattern structures and their projections. In: ICCS. Volume 2120 of LNCS. Springer Berlin Heidelberg (2001) 129–142
8. Pernelle, N., Rousset, M.C., Soldano, H., Ventos, V.: Zoom: a nested galois lattices-based system for conceptual clustering. Exp. Theor. Artif. Intell. (2002) 157–187
9. Soldano, H., Ventos, V.: Abstract concept lattices. In: Formal Concept Analysis. Volume 6628 of LNCS. Springer Berlin Heidelberg (2011) 235–250
10. Buzmakov, A., Kuznetsov, S., Napoli, A.: Revisiting pattern structure projections. In: Formal Concept Analysis. Springer International Publishing (2015) 200–215
11. Stumme, G.: Attribute exploration with background implications and exceptions. In: Data Analysis and Information Systems. Springer (1996) 457–469
12. Ganter, B.: Attribute exploration with background knowledge. Theoretical Computer Science (1999) 215 – 233