Formal Concept Analysis Constrained by Attribute-Dependency Formulas

Radim Bělohlávek and Vladimír Sklenář

Dept. Computer Science, Palacký University, Tomkova 40, CZ-779 00 Olomouc, Czech Republic
{radim.belohlavek, vladimir.sklenar}@upol.cz

Abstract. An important topic in formal concept analysis is to cope with a possibly large number of formal concepts extracted from formal context (input data). We propose a method to reduce the number of extracted formal concepts by means of constraints expressed by particular formulas (attribute-dependency formulas, ADF). ADF represent a form of dependencies specified by a user expressing relative importance of attributes. ADF are considered as additional input accompanying the formal context $\langle X, Y, I\rangle$. The reduction consists in considering formal concepts which are compatible with a given set of ADF and leaving out noncompatible concepts. We present basic properties related to ADF, an algorithm for generating the reduced set of formal concepts, and demonstrating examples.

1 Preliminaries and Problem Setting

We refer to [6] (see also [14]) for background information in formal concept analysis (FCA). We denote a formal context by $\langle X, Y, I\rangle$, i.e. $I \subseteq X \times Y$ (object-attribute data table, objects $x \in X$, attributes $y \in Y$); the concept deriving operators by $^{\uparrow}$ and $^{\downarrow}$, i.e. for $A \subseteq X$, $A^{\uparrow} = \{y \in Y \mid \text{ for each } x \in A : \langle x, y\rangle \in I\}$ and dually for $^{\downarrow}$; a concept lattice of $\langle X, Y, I\rangle$ by $\mathcal{B}(X, Y, I)$, i.e. $\mathcal{B}(X, Y, I) = \{\langle A, B\rangle \in 2^X \times 2^Y \mid A^{\uparrow} = B,\ B^{\downarrow} = A\}$.

An important aspect of FCA is a possibly large number of formal concepts in $\mathcal{B}(X, Y, I)$. Very often, the formal concepts contain those which are in a sense not interesting for the expert. In this paper, we present a way to naturally reduce the number of formal concepts extracted from data by taking into account information additionally supplied to the input data table (formal context). We consider a particular form of the additional information, namely, a form of particular attribute dependencies expressed by (logical) formulas that can be supplied by an expert/user. The primary interpretation of the dependencies is to express a kind of relative importance of attributes. We introduce the notion of a formal concept compatible with the attribute dependencies. The main gain of considering only compatible formal concepts and disregarding formal concepts which are not compatible is the reduction of the number of resulting formal concepts. This leads to a more comprehensible structure of formal concepts (clusters) extracted from the input data. We present basic theoretical results, an algorithm for generating compatible formal concepts, and illustrate our approach by examples.

2 Constraints by Attribute Dependencies

2.1 Motivation

When people categorize objects by means of their attributes, they naturally take into account the importance of attributes. Usually, attributes which are less important are not used to form large categories (clusters, concepts). Rather, less important attributes are used to make a finer categorization within a larger category. For instance, consider a collection of certain products offered on a market, e.g. home appliances. When categorizing home appliances, one may consider several attributes like price, the purpose of the appliance, the intended placement of the appliance (kitchen appliance, bathroom appliance, office appliance, etc.), power consumption, color, etc. Intuitively, when forming appliance categories, one picks the most important attributes and forms the general categories like "kitchen appliances", "office appliances", etc. Then, one may use the less important attributes (like "price $\leq$ \$10", "price between \$15–\$40", "price $>$ \$100", etc.) and form categories like "kitchen appliance with price between \$15–\$40". Within this category, one may further form finer categories distinguished by color. This pattern of forming categories follows the rule that when an attribute y is to belong to a category, the category must contain an attribute which determines a more important characteristic of the attribute (like "kitchen appliance" determines the intended placement of the appliance). This must be true for all the characteristics that are more important than y. In this sense, the category "red appliance" is not well-formed since color is considered less important than price and the category "red appliance" does not contain any information about the price. Which attributes and characteristics are considered more important depends on the particular purpose of categorization. In the above example, it may well be the case that price be considered more important than the intended placement. Therefore, the information about the relative importance of the attributes is to be supplied by an expert (the person who determines the purpose of the categorization). Once the information has been supplied, it serves as a constraint for the formation of categories. In what follows, we propose a formal approach to the treatment of the above-described constraints to formation of categories.

2.2 Constraints by Attribute-Dependency Formulas

Consider a formal context $\langle X, Y, I\rangle$. We consider constraints expressed by formulas of the form

$$y \sqsubseteq y_1 \sqcup \cdots \sqcup y_n. \tag{1}$$

Formulas (1) will be called AD-formulas (attribute-dependency formulas). The set of all AD-formulas will be denoted by ADF. Let now $\mathcal{C} \subseteq ADF$ be a set of AD-formulas.

Definition 1. *A formal concept $\langle A, B\rangle$ satisfies an AD-formula (1) if we have that*

$$\textit{if } y \in B \textit{ then } y_1 \in B \textit{ or } \cdots \textit{ or } y_n \in B.$$

Remark 1. More generally, we could consider formulas $l(y) \sqsubseteq l(y_1) \sqcup \cdots \sqcup l(y_n)$ where $l(z)$ is either z or $\overline{z}$. For instance, $y \sqsubseteq \overline{y_1}$ would be satisfied by $\langle A, B\rangle$ if whenever $y \in B$ then none of $x \in A$ has y_1. For the purpose of our paper, however, we consider only (1).

The fact that $\langle A, B\rangle \in \mathcal{B}(X, Y, I)$ satisfies an AD-formula φ is denoted by $\langle A, B\rangle \models \varphi$. Therefore, $\models$ is the basic satisfaction relation (being a model) between the set $\mathcal{B}(X, Y, I)$ of all formal concepts (models, structures) and the set *ADF* of all AD-formulas (formulas). As usual, $\models$ induces two mappings, $\mathrm{Mod} : 2^{ADF} \to 2^{\mathcal{B}(X,Y,I)}$ assigning a subset

$$\mathrm{Mod}(\mathcal{C}) = \{\langle A, B\rangle \in \mathcal{B}(X, Y, I) \mid \langle A, B\rangle \models \varphi \text{ for each } \varphi \in \mathcal{C}\}$$

to a set $\mathcal{C} \subseteq ADF$ of AD-formulas, and $\mathrm{Fml} : 2^{\mathcal{B}(X,Y,I)} \to 2^{ADF}$ assigning a subset

$$\mathrm{Fml}(U) = \{\varphi \in ADF \mid \langle A, B\rangle \models \varphi \text{ for each } \langle A, B\rangle \in U\}$$

to a subset $U \subseteq \mathcal{B}(X, Y, I)$. The following result is immediate [12].

Theorem 1. *The mappings* Mod *and* Fml *form a Galois connection between ADF and $\mathcal{B}(X, Y, I)$. That is, we have*

$$\mathcal{C}_1 \subseteq \mathcal{C}_2 \textit{ implies } \mathrm{Mod}(\mathcal{C}_2) \subseteq \mathrm{Mod}(\mathcal{C}_1), \tag{2}$$
$$\mathcal{C} \subseteq \mathrm{Fml}(\mathrm{Mod}(\mathcal{C})), \tag{3}$$
$$U_1 \subseteq U_2 \textit{ implies } \mathrm{Fml}(U_2) \subseteq \mathrm{Fml}(U_1), \tag{4}$$
$$U \subseteq \mathrm{Mod}(\mathrm{Fml}(U)). \tag{5}$$

for any $\mathcal{C}, \mathcal{C}_1, \mathcal{C}_2 \subseteq ADF$, and $U, U_1, U_2 \subseteq \mathcal{B}(X, Y, I)$.

Definition 2. *For $\mathcal{C} \subseteq ADF$ we put*

$$\mathcal{B}_{\mathcal{C}}(X, Y, I) = \mathrm{Mod}(\mathcal{C})$$

and call it the constrained (by $\mathcal{C}$) concept lattice *induced by $\langle X, Y, I\rangle$ and $\mathcal{C}$.*

For simplicity, we also denote $\mathcal{B}_{\mathcal{C}}(X, Y, I)$ by $\mathcal{B}_{\mathcal{C}}$. That is, $\mathcal{B}_{\mathcal{C}}(X, Y, I)$ is the collection of all formal concepts from $\mathcal{B}(X, Y, I)$ which satisfy each AD-formula from $\mathcal{C}$ (satisfy all constraints from $\mathcal{C}$).

The following is immediate.

Theorem 2. *$\mathcal{B}_{\mathcal{C}}(X, Y, I)$ is a partially ordered subset of $\mathcal{B}(X, Y, I)$ which is bounded from below. Moreover, if $\mathcal{C}$ does not contain an AD-formula* (1) *such that y is shared by all objects from X and none of $y_1, \ldots, y_n$ is shared by all objects, then $\mathcal{B}_{\mathcal{C}}(X, Y, I)$ is bounded from above.*

Proof. Obviously, $\langle Y^{\downarrow}, Y\rangle$ is the least formal concept from $\mathcal{B}(X,Y,I)$ and it is compatible with each AD-formula. Therefore, $\langle Y^{\downarrow}, Y\rangle$ bounds $\mathcal{B}_{\mathcal{C}}(X,Y,I)$ from below. Furthermore, if there is no AD-formula (1) with the above-mentioned properties then $\langle X, X^{\uparrow}\rangle$ is the upper bound of $\mathcal{B}_{\mathcal{C}}(X,Y,I)$ since in this case $\langle X, X^{\uparrow}\rangle$ clearly satisfies $\mathcal{C}$.

Remark 2. Note that the condition guaranteeing that $\mathcal{B}_{\mathcal{C}}(X,Y,I)$ is bounded from above is usually satisfied. Namely, in most cases, there is no object satisfying all attributes and so $X^{\uparrow} = \emptyset$ in which case the condition is fulfilled.

Let us now consider AD-formulas of the form

$$y \sqsubseteq y'. \tag{6}$$

Clearly, (6) is a particular form of (1) for $n = 1$. Constraints equivalent to (6) were considered in [1, 2], see also [9] for a somewhat different perspective. In [1, 2], constraints are considered in the form of a binary relation R on a set of attributes (in [1]) or objects (in [2]). On the attributes, R might be a partial order expressing importance of attributes; on the objects, R might be an equivalence relation expressing some partition of objects. Restricting ourselves to AD-formulas (6), $\mathcal{B}_{\mathcal{C}}(X,Y,I)$ is itself a complete lattice:

Theorem 3. *Let $\mathcal{C}$ be a set of AD-formulas of the form* (6)*. Then $\mathcal{B}_{\mathcal{C}}(X,Y,I)$ is a complete lattice which is a $\bigvee$-sublattice of $\mathcal{B}(X,Y,I)$.*

Proof. Since $\mathcal{B}_{\mathcal{C}}(X,Y,I)$ is bounded from below (Theorem 2), it suffices to show that $\mathcal{B}_{\mathcal{C}}(X,Y,I)$ is closed under suprema in $\mathcal{B}(X,Y,I)$, i.e. that for $\langle A_j, B_j\rangle \in \mathcal{B}_{\mathcal{C}}(X,Y,I)$ we have $\langle(\cap_j B_j)^{\downarrow}, \cap_j B_j\rangle \in \mathcal{B}_{\mathcal{C}}(X,Y,I)$. This can be directly verified.

One can show that $\mathcal{B}_{\mathcal{C}}(X,Y,I)$ in Theorem 3 need not be a $\bigwedge$-sublattice of $\mathcal{B}(X,Y,I)$. Note that $\langle A,B\rangle \models (y \sqsubseteq y')$ says that B contains y' whenever it contains y. Then, $\langle A,B\rangle \models \{y \sqsubseteq y', y' \sqsubseteq y\}$ if either both y and y' are in B or none of y and y' is in B. This seems to be interesting particularly in the dual case, i.e. when considering constraints on objects, to select only formal concepts which do not separate certain groups of objects (for instance, the groups may form a partition known from outside or generated from the formal context).

In the rest of this section we briefly discuss selected topics related to constraints by AD-formulas. Due to the limited space, we omit details.

2.3 Expressive Power of AD-Formulas

An attribute may occur on left hand-side of several AD-formulas of $\mathcal{C}$. For example, we may have $y \sqsubseteq y_1 \sqcup y_2$ and $y \sqsubseteq y_3 \sqcup y_4$. Then, for a formal concept $\langle A,B\rangle$ to be compatible, it has to satisfy the following: whenever $y \in B$ then it must be the case that $y_1 \in B$ or $y_2 \in B$, and $y_3 \in B$ or $y_4 \in B$. Therefore, it is tempting to allow for expressions of the form

$$y \sqsubseteq (y_1 \sqcup y_2) \sqcap (y_3 \sqcup y_4)$$

with the intuitively clear meaning of compatibility of a formal concept and a formula of this generalized form. Note that a particular form is also e.g. $y \sqsubseteq y_2 \sqcap y_3$. One may also want to extend this form to formulas containing disjunctions of conjunctions, e.g.

$$y \sqsubseteq (y_1 \sqcap y_2) \sqcup (y_3 \sqcap y_4).$$

In general, one may consider formulas of the form

$$y \sqsubseteq t(y_1, \dots, y_n) \tag{7}$$

where $t(y_1, \dots, y_n)$ is a term over Y defined by: (i) each attribute $y \in Y$ is a term; (ii) if t_1 and t_2 are terms then $(t_1 \sqcup t_2)$ and $(t_1 \sqcap t_2)$ are terms. Then for a set $\mathcal{D}$ of formulas of the form (7), $\mathcal{B}_{\mathcal{D}}(X, Y, I)$ has the obvious meaning (formal concepts from $\mathcal{B}(X, Y, I)$ satisfying all formulas from $\mathcal{D}$). The following assertion shows that with respect to the possibility of expressing constraints, we do not gain anything new by allowing formulas (7).

Theorem 4. *For each set $\mathcal{D}$ of formulas* (7) *there is a set $\mathcal{C}$ of AD-formulas such that $\mathcal{B}_{\mathcal{D}}(X, Y, I) = \mathcal{B}_{\mathcal{C}}(X, Y, I)$.*

Proof. It is tedious but straightforward to show that each formula φ of the form (7) can be transformed to an equivalent formula of the form $y \sqsubseteq D_1 \sqcap \dots \sqcap D_m$ where each D_k is of the form $y_{k,1} \sqcup \dots \sqcup y_{k,l_k}$. Now, $y \sqsubseteq D_1 \sqcap \dots \sqcap D_m$ is equivalent to the set $AD(\varphi) = \{y \sqsubseteq D_i \mid i = 1, \dots, m\}$ of AD-formulas. Therefore, $\mathcal{D}$ is equivalent to $\mathcal{C} = \bigcup_{\varphi \in \mathcal{D}} AD(\varphi)$ in that $\mathcal{B}_{\mathcal{D}}(X, Y, I) = \mathcal{B}_{\mathcal{C}}(X, Y, I)$. An easy way to see this is to look at $\sqsubseteq, \sqcup, \sqcap$ as propositional connectives of implication, disjunction, and conjunction, respectively, and to observe that a formula φ of the form (7) is satisfied by a formal concept $\langle A, B \rangle$ iff the propositional formula corresponding to φ is true under a valuation $v : Y \to \{0, 1\}$ defined by $v(y) = 1$ if $y \in B$ and $v(y) = 0$ if $y \notin B$.

We will demonstrate the expressive capability of AD-formulas in Section 3.

2.4 Entailment of AD-Formulas

We now focus on the notion of entailment of AD-formulas. To this end, we extend in an obvious way the notion of satisfaction of an AD-formula. We say that a subset $B \subseteq Y$ (not necessarily being an intent of some formal concept) satisfies an AD-formula φ of the form (1) if we have that if $y \in B$ then some of $y_1, \dots, y_n$ belongs to B as well and denote this fact by $B \models \varphi$ (we use $B \models \{\varphi_1, \dots, \varphi_n\}$ in an obvious sense).

Definition 3. *An AD-formula φ (semantically) follows from a set $\mathcal{C}$ of AD-formulas if for each $B \subseteq Y$ we have that if $B \models \mathcal{C}$ (B satisfies each formula from $\mathcal{C}$) then $B \models \varphi$.*

We are going to demonstrate an interesting relationship between AD-formulas and so-called attribute implications which are being used in formal concept analysis and have a strong connection to functional dependencies in databases, see [6, 10]. Recall that an attribute implication is an expression of the form $A \Rightarrow B$ where $A, B \subseteq Y$. We say that $A \Rightarrow B$ is satisfied by $C \subseteq Y$ (denoted by $C \models A \Rightarrow B$) if $B \subseteq C$ whenever $A \subseteq C$ (this obviously extends to $\mathcal{M} \models T$ for a set $\mathcal{M}$ of subsets of Y and a set T of attribute implications). The connection between AD-formulas and attribute implications is the following.

Lemma 1. *For a set $B \subseteq Y$ we have*

$$B \models y \sqsubseteq y_1 \sqcup \cdots \sqcup y_n \qquad \text{iff} \qquad \overline{B} \models \{y_1, \ldots, y_n\} \Rightarrow y$$

where $\overline{B} = Y - B$. Furthermore,

$$B \models \{y_1, \ldots, y_n\} \Rightarrow \{z_1, \ldots, z_m\} \qquad \text{iff}$$
$$\overline{B} \models z_i \sqsubseteq y_1 \sqcup \cdots \sqcup y_n \quad \text{for each } i = 1, \ldots, m.$$

Proof. The assertion follows from definition by a moment's reflection.

Now, this connection can be used to reducing the notion of semantical entailment of AD-formulas to that of entailment of attribute implications which is well studied and well known [6, 10]. Recall that an attribute implication φ (semantically) follows from a set $\mathcal{C}$ of attribute implications ($\mathcal{C} \models \varphi$) if φ is true in each $B \subseteq Y$ which satisfies each attribute implication from $\mathcal{C}$. For an AD-formula $\varphi = y \sqsubseteq y_1 \sqcup \cdots \sqcup y_n$, denote by $I(\varphi)$ the attribute implication $\{y_1, \ldots, y_n\} \Rightarrow y$. Conversely, for an attribute implication $\varphi = \{y_1, \ldots, y_n\} \Rightarrow \{z_1, \ldots, z_m\}$, denote by $A(\varphi)$ the set $\{z_i \sqsubseteq y_1 \sqcup \cdots \sqcup y_n \mid i = 1, \ldots, m\}$ of AD-formulas. Then it is immediate to see that we have the following "translation rules":

Lemma 2. *(1) Let φ, φ_j, $(j \in J)$ be AD-formulas. Then $\{\varphi_j \mid j \in J\} \models \varphi$ (entailment of AD-formulas) iff $\{I(\varphi_j) \mid j \in J\} \models I(\varphi)$ (entailment of attribute implications). (2) Let φ, φ_j, $(j \in J)$ be attribute implications. Then $\{\varphi_j \mid j \in J\} \models \varphi$ (entailment of attribute implications) iff $\cup_{j \in J} A(\varphi_j) \models A(\varphi)$ (entailment of AD-formulas).*

Lemma 2 gives a possibility to answer important problems related to entailment like closedness, completeness, (non)redundancy, being a base, etc. Due to the limited space, we leave a detailed discussion to our forthcoming paper.

2.5 Trees from Concept Lattices

A particularly interesting structure of clusters in data is that of a tree, see [3, 5]. In a natural way, trees can be extracted from concept lattices by AD-formulas. Here, we present a criterion under which $\mathcal{B}_{\mathcal{C}}(X, Y, I)$ is a tree. Since $\mathcal{B}_{\mathcal{C}}(X, Y, I)$ has always the least element (see Theorem 2), we will call $\mathcal{B}_{\mathcal{C}}(X, Y, I)$ a tree if $\mathcal{B}_{\mathcal{C}}(X, Y, I) - \{\langle Y^{\downarrow}, Y \rangle\}$ (deleting the least element) is a tree.

Theorem 5. *If for each $y_1, y_2 \in Y$ which are not disjoint (i.e. $\{y_1\}^{\downarrow} \cap \{y_2\}^{\downarrow} \neq \emptyset$) $\mathcal{C}$ contains an AD-formula $y_1 \sqsubseteq \cdots \sqcup y_2 \sqcup \cdots$ or $y_2 \sqsubseteq \cdots \sqcup y_1 \sqcup \cdots$ such that the attributes on the right hand-side of each the formulas are pairwise disjoint then $\mathcal{B}_{\mathcal{C}}(X, Y, I)$ is a tree.*

Proof. If $\mathcal{B}_{\mathcal{C}}\,(X,Y,I)$ is not a tree, there are distinct and noncomparable $\langle A_1,B_1\rangle$, $\langle A_2,B_2\rangle \in \mathcal{B}_{\mathcal{C}}\,(X,Y,I)$ such that $A_1 \cap A_2 \neq Y^{\downarrow}$ (their meet in $\mathcal{B}_{\mathcal{C}}\,(X,Y,I)$) is greater than the least formal concept $\langle Y^{\downarrow}, Y\rangle$. Then there are $y_1 \in B_1 - B_2$ and $y_2 \in B_2 - B_1$. But y_1 and y_2 cannot be disjoint (otherwise $A_1 \cap A_2 = \emptyset$ which is not the case). By assumption (and without the loss of generality), $\mathcal{C}$ contains an AD-formula $y_1 \sqsubseteq \cdots \sqcup y_2 \sqcup \cdots$. Now, since there is some $x \in A_1 \cap A_2$, we have $\langle x, y_1\rangle \in I$ and $\langle x, y_2\rangle \in I$. Since $\langle A_1, B_1\rangle$ satisfies $y_1 \sqsubseteq \cdots \sqcup y_2 \sqcup \cdots$, B_1 must contain some y' which appears on the right hand-side of the AD-formula, from which we get $\langle x, y'\rangle$. But $\langle x, y_2\rangle \in I$ and $\langle x, y'\rangle \in I$ contradict the disjointness of y_2 and y'.

Remark 3. Note that the assumption of Theorem 5 is satisfied in the following situation: Attributes from Y are partitioned into subsets $Y_1, \ldots, Y_n$ of Y such that each $Y_i = \{y_{i,1}, \ldots, y_{i,n_i}\}$ corresponds to some higher-level attribute. E.g., Y_i may correspond to color and may contain attributes red, gree, blue, ... Then, we linearly order Y_i's, e.g. by $Y_1 < \ldots Y_n$ and for each $i < j$ add a set $Y_i \sqsubseteq Y_j$ of AD-formulas of the form $y_{i,j} \sqsubseteq y_{j,1} \sqcup \cdots \sqcup y_{j,n_j}$ for each $y_{i,j} \in Y_i$. In fact, we may add only $Y_i \sqsubseteq Y_{i+1}$ and omit the rest with the same restriction effect. Then the assumptions of Theorem 5 are met. Such a situation occurs when one linearly orders higher-level attributes like color < price < manufacturer and want to see the formal concepts respecting this order. The just described ordering of attributes is typical of so-called decision trees [13].

Remark 4. Note that formulas $y_1 \sqsubseteq \cdots \sqcup y_2 \sqcup \cdots$ and $y_2 \sqsubseteq \cdots \sqcup y_1 \sqcup \cdots$ in Theorem 5 need not belong to $\mathcal{C}$. It is sufficient if they are entailed by $\mathcal{C}$, i.e. if $\mathcal{C} \models y_1 \sqsubseteq \cdots \sqcup y_2 \sqcup \cdots$ or $\mathcal{C} \models y_2 \sqsubseteq \cdots \sqcup y_1 \sqcup \cdots$.

2.6 Algorithm for Computing $\mathcal{B}_{\mathcal{C}}\,(X,Y,I)$

In this section we present an algorithm for computing $\mathcal{B}_{\mathcal{C}}\,(X,Y,I)$. The algorithm computes both the formal concepts and their ordering, and is a modification of the incremental algorithms (see [8]), particularly of [11]. The purpose of our algorithm is to compute $\mathcal{B}_{\mathcal{C}}\,(X,Y,I)$ without the need to compute the whole $\mathcal{B}\,(X,Y,I)$ and then test which formal concepts $\langle A,B\rangle \in \mathcal{B}\,(X,Y,I)$ satisfy $\mathcal{C}$. For the reader's convenience, we list the whole pseudocode. For explaining of the idea of the background algorithm we refer to [11]. In what follows, we comment only on the modification related to taking into account the constraining AD-formulas. In addition to `CreateLatticeIncrementally` (computes the constrained lattice) and `GetMaximalConcept` (auxiliary function) which are the same as in [11], our modification consists in introducing a new function `GetValidSubIntent` and a new function `ADFAddIntent` which is a modified version of `AddIntent` from [11].

The algorithm is a kind of so-called incremental algorithms. The main procedure, called `CreateLatticeIncrementally`, calls for each object $x \in X$ recursive function `ADFAddIntent`. `ADFAddIntent` checks whether the intent of the input object x satisfies AD-formulas from $\mathcal{C}$. If yes, it passes the intent. If not, it passes the largest subset of the intent which satisfies $\mathcal{C}$. The largest subset B is obtained by `GetValidSubIntent` and need not be an intent. Nevertheless, `ADFAddIntent`, in the further processing, can create new intents from B. This way, “false” concepts may be created. Therefore, after processing of all the objects, we need to check and remove the “false” concepts. The algorithm plus its illustration follow.

```
Procedure CreateLatticeIncrementally (X,Y,I)
 BottomConcept := (Y',Y)
 L := {BottomConcept}
 For each x in X
  ObjectConcept := AddIntent(x', BottomConcept, L)
  Add x to the extent of ObjectConcept and all concepts above
 End For

Function ADFAddIntent(inputIntent, generator, L)
 if IsValidIntent(inputIntent)
     intent := inputIntent
 else
     intent := GetValidSubIntent(inputIntent)
 End If
     GeneratorConcept = GetMaximalConcept(intent, generator,L)
     If GeneratorConcept.intent = intent
      Return GeneratorConcept
     End If
     newParents := Empty Set
     GeneratorParents = GetParents(GeneratorConcept, L)
     For each Candidate in GeneratorParents
        if Candidate.Intent Is Not SubSet intent
         Candidate:=
         ADFAddIntent(Intersection(Candidate.Intent,intent),
                               Candidate, L)
        End if
          addParent = true
          for each Parent in NewParents
             if Candidate.Intent Is Subset Parent.Intent
                addParent := false
                Exit For
             Else If Parent.Intent Is Subset Candidate.Intent
                Remove Parent from newParents
             End If
          End For
       if addParent
         Add candidate to newParents
       End If
        End For
     newConcept := (GeneratorConcept.Extent,intent)
     Add NewConcept to L
```

```
    For each Parent in newParents
        RemoveLink(Parent, GeneratorConcept, L)
        SetLink(Parent, NewConcept, L)
    End For
    SetLink(NewConcept, GeneratorConcept, L)
Return NewConcept
```

```
Function GetValidSubIntent(intent, L)
 notValid := true
 While notValid and intent <> EmptySet
     conflictADF := GetConflictADF (intent, L)
   For each adf in conflictADF
     Remove adf.LeftSide from intent
   End For
   notValid := IsValidIntent(intent)
   End If
 End While
return intent
```

```
Function GetMaximalConcept(intent, GeneratorConcept, L)
 parentIsMaximal := true
 While parentIsMaximal
  parentIsMaximal := false
  Parents = GetParents(GeneratorConcept, L)
  For each Parent in Parents
   If intent is subset Parent.Intent
    GeneratorConcept := Parent
    parentIsMaximal := true
    Exit For
   End If
  End For
 End While
 return GeneratorConcept
```

We now illustrate the algorithm on an example which is a modification of an example from [8].

First, we add object 6 with attributes b, c, f, i to (already computed) concept lattice depicted in Fig. 1. First, if we do not apply any constraints, we obtain 4 new concepts and get the concept lattice in Fig. 2.

Second, we add 6 under an AD-formula $f \sqsubseteq h \sqcup i$ (note that all concepts from the concept lattice in Fig. 1 satisfy this AD-formula). In this case, the intent of 6 satisfies $f \sqsubseteq h \sqcup i$. As a result, we get the constrained concept lattice in Fig. 3.

Third, we add object 6 under an AD-formula $i \sqsubseteq a$ (note again that all concepts from the concept lattice in Fig. 1 satisfy this AD-formula). In this case, the intent of 6 does not satisfy $i \sqsubseteq a$. As a result, we get the constrained concept lattice in Fig. 4.

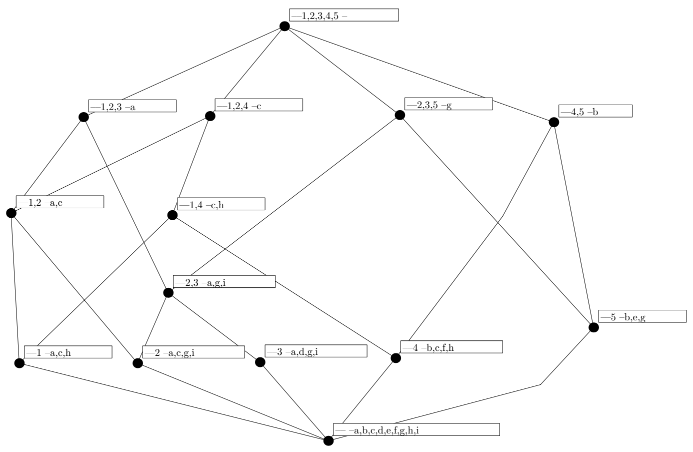

Fig. 1. A concept lattice to which we wish to add object 6 having attributes b, c, f, i, under different constraints

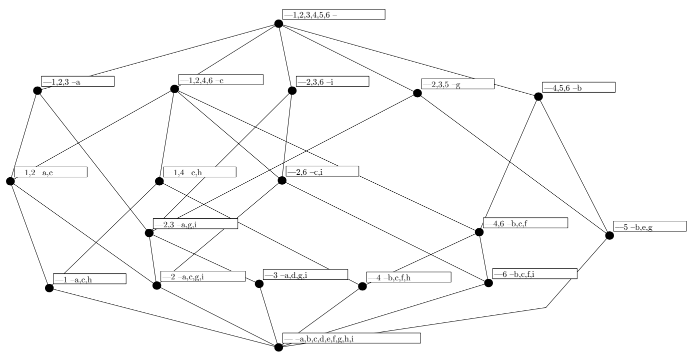

Fig. 2. Adding object 6 to concept lattice from Fig. 1 with no constraints

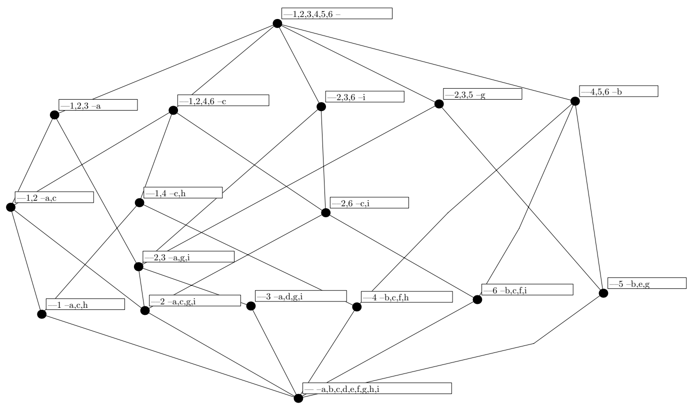

Fig. 3. Adding object 6 to concept lattice from Fig. 1 under constraint f $\sqsubseteq$ h $\sqcup$ i

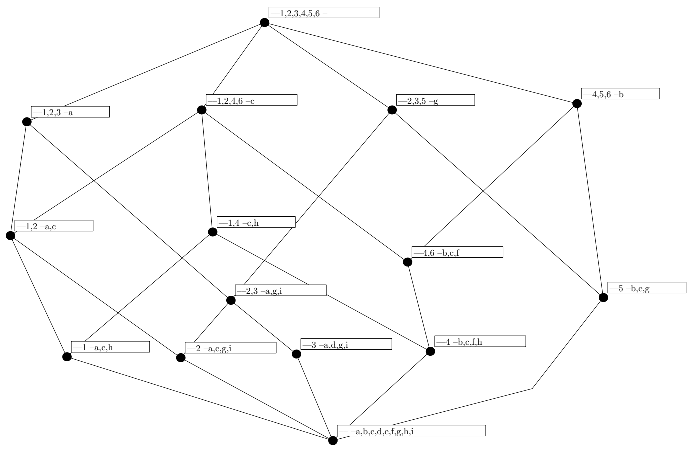

Fig. 4. Adding object 6 to concept lattice from Fig. 1 under constraint i $\sqsubseteq$ a

3 Examples

We now present illustrative examples. We use Hasse diagrams and label the nodes corresponding to formal concepts by boxes containing concept descriptions. For example, $(\{1,3,7\},\{3,4\})$ is a concept with extent $\{1,3,7\}$ and intent $\{3,4\}$. Consider a formal context described in Tab. 1. The context represents data about eight car models (1–8) and their selected attributes (1: diesel engine,. . . , 8: ABS).

Table 1. Formal context given by cars and their properties

	1	2	3	4	5	6	7	8
car 1	1	0	1	0	0	1	0	1
car 2	1	0	1	0	1	1	0	1
car 3	0	1	1	0	0	0	0	1
car 4	0	1	0	1	1	0	0	0
car 5	0	1	1	0	1	1	0	0
car 6	0	1	0	1	0	1	1	0
car 7	0	1	0	1	1	1	1	1
car 8	0	1	0	1	0	0	0	1

attributes: 1 - diesel engine, 2 - gasoline engine, 3 - sedan, 4 - hatchback, 5 - air-conditioning, 6 - airbag, 7 - power stearing, 8 - ABS

The attributes can be partitioned into three groups: $\{1,2\}$ (engine type), $\{3,4\}$ (car type), $\{5,6,7,8\}$ (equipment). The concept lattice $\mathcal{B}(X,Y,I)$ corresponding to formal concept $\langle X,Y,I\rangle$ contains 27 formal concepts and is depicted in Fig. 5. The formal concepts of $\mathcal{B}(X,Y,I)$ represent, in the sense of FCA, all meaninfull concepts-clusters present in the data. Suppose we want to use a (part

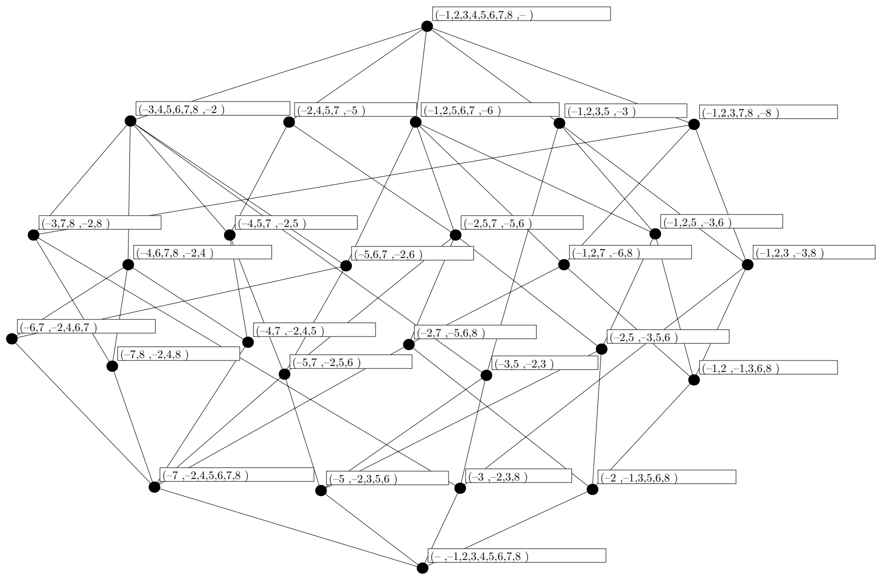

Fig. 5. Concept lattice corresponding to the context from Tab. 1

of a) concept lattice to provide a conceptual view of the data table and suppose we know that the customers find most important the type of engine (diesel, gasoline) and then the type of the bodywork. Such a situation is described by a set $\mathcal{C}$ of AD-formulas from (8).

$$\begin{aligned}
\text{air-conditioning} &\sqsubseteq \text{hatchback} \sqcup \text{sedan} \\
\text{power stearing} &\sqsubseteq \text{hatchback} \sqcup \text{sedan} \\
\text{airbag} &\sqsubseteq \text{hatchback} \sqcup \text{sedan} \\
\text{ABS} &\sqsubseteq \text{hatchback} \sqcup \text{sedan} \\
\text{hatchback} &\sqsubseteq \text{gasoline engine} \sqcup \text{diesel engine} \\
\text{sedan} &\sqsubseteq \text{gasoline engine} \sqcup \text{diesel engine}
\end{aligned} \tag{8}$$

The corresponding constrained $\mathcal{B}_{\mathcal{C}}(X, Y, I)$ contains 13 formal concepts and is depicted in Fig. 6. Second, consider a set $\mathcal{C}$ of AD-formulas (9). Contrary to the previous example, the importance of the type of a bodywork and the type of the engine are reversed.

$$\begin{aligned}
\text{air-conditioning} &\sqsubseteq \text{diesel engine} \sqcup \text{gasoline engine} \\
\text{power stearing} &\sqsubseteq \text{diesel engine} \sqcup \text{gasoline engine} \\
\text{airbag} &\sqsubseteq \text{diesel engine} \sqcup \text{gasoline engine} \\
\text{ABS} &\sqsubseteq \text{diesel engine} \sqcup \text{gasoline engine} \\
\text{gasoline engine} &\sqsubseteq \text{hatchback} \sqcup \text{sedan} \\
\text{diesel engine} &\sqsubseteq \text{hatchback} \sqcup \text{sedan}
\end{aligned} \tag{9}$$

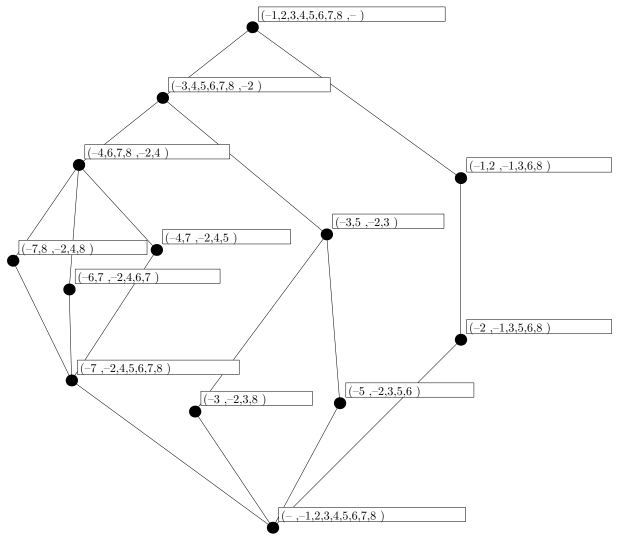

Fig. 6. Concept lattice from Fig. 5 constrained by AD-formulas 8

The corresponding constrained $\mathcal{B}_{\mathcal{C}}\left(X,Y,I\right)$ contains 13 formal concepts and is depicted in Fig. 7. We can see that the bottom parts in Fig. 6 and Fig. 7 are the same. The lattices differ according to the selection of importance of the attributes.

Third, suppose the customer changes the preferences and finds the most important car property to be safety and requires ABS and airbag. The situation is described by a set $\mathcal{C}$ of AD-formulas from 10.

$$\begin{aligned} \text{air-conditioning} &\sqsubseteq \text{diesel engine} \sqcup \text{gasoline engine} \\ \text{power stearing} &\sqsubseteq \text{diesel engine} \sqcup \text{gasoline engine} \\ \text{gasoline engine} &\sqsubseteq \text{hatchback} \sqcup \text{sedan} \\ \text{diesel engine} &\sqsubseteq \text{hatchback} \sqcup \text{sedan} \\ \text{sedan} &\sqsubseteq \text{ABS} \\ \text{hatchback} &\sqsubseteq \text{ABS} \\ \text{ABS} &\sqsubseteq \text{airbag} \\ \text{airbag} &\sqsubseteq \text{ABS} \end{aligned} \tag{10}$$

The resulting $\mathcal{B}_{\mathcal{C}}\left(X,Y,I\right)$ corresponding to (10) contains 6 formal concepts and is depicted in Fig. 8. In general, if an attribute $y' \in Y$ is required, it is sufficient to have $\mathcal{C}$ such that $\mathcal{C} \models \{y \sqsubseteq y'\}$ for each $y \in Y$. In particular, for $\mathcal{C} = \{y \sqsubseteq y' \mid y \in Y\}$ we have $\mathcal{B}_{\mathcal{C}}\left(X,Y,I\right) = \{\langle A,B\rangle \in \mathcal{B}\left(X,Y,I\right) \mid \langle \{y'\}^{\downarrow}, \{y'\}^{\downarrow\uparrow}\rangle \leq \langle A,B\rangle\}$, i.e. $\mathcal{B}_{\mathcal{C}}\left(X,Y,I\right)$ is a main filter in $\mathcal{B}\left(X,Y,I\right)$ corresponding to attribute concept of y'.

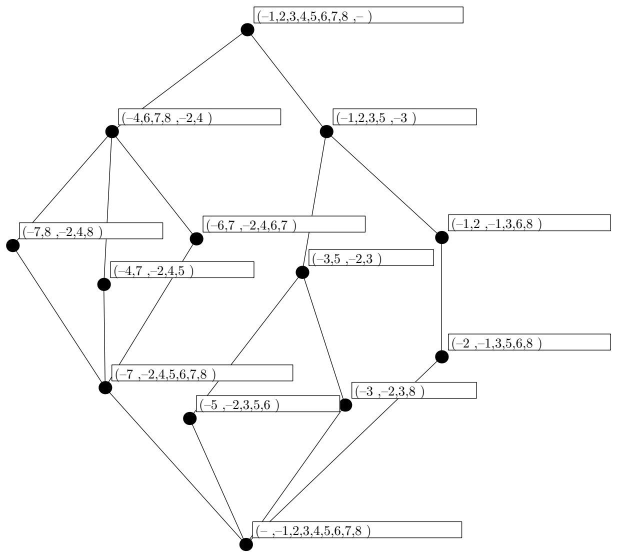

Fig. 7. Concept lattice from Fig. 5 constrained by AD-formulas 9

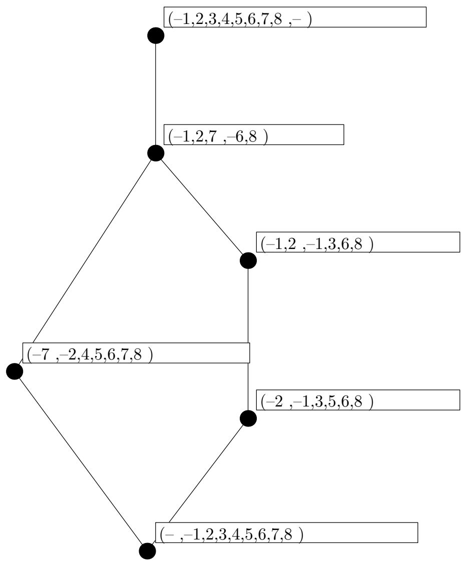

Fig. 8. Concept lattice corresponding to AD-formulas 10

4 Future Research

In the next future, we focus on the following:

– As one of the referees pointed out, AD-formulas are what is called surmise relationships in [4]. In [7], even more general formulas (clauses) are studied. A future research will be focused on constraints by more general formulas, e.g. to allow negations (see Section 2.2), and on studying relationships to earlier work on formulas over attributes, see [7] and the papers cited therein.

– Entailment. In Section 2.4, we showed basic observations which make it possible to reduce the problems related to entailment of AD-formulas to the corresponding problems of entailment of attribute implications which are well described. In our forthcoming paper, we elaborate more on this.
– Algorithms. We will study further possibilities to generate concepts satisfying the constraints directly, see Section 2.6.
– Experiments with large datasets. In a preliminary study we used a database of mushrooms avaliable at UCI KDD Archive (http://kdd.ics.uci.edu/). We transformed the database to a formal context with 8124 objects and 119 attributes. The corresponding concept lattice contains 238710 formal concepts and took about 10 minutes to compute it. We run experiments with several sets of AD-formulas which seemed to express natural constraints. The resulting constrained concept lattices were considerably smaller (from tens to thousands of formal concepts, depending on the sets of AD-formulas). The computation of the constrained concept lattices took a correspondingly smaller amount of time (from 2 seconds up). We will run further experiments on this database, possibly with an expert in the field and compare our results with other clustering methods.

Acknowledgment. Radim Bělohlávek gratefully acknowledges support by grant No. 201/02/P076 of the Grant Agency of the Czech Republic. Also, we acknowledge support by grant No. 1ET101370417 of the GA AV CR.

References

1. Bělohlávek R., Sklenář V., Zacpal J.: Formal concept analysis with hierarchically ordered attributes. *Int. J. General Systems* **33**(4)(2004), 283–294.
2. Bělohlávek R., Sklenář V., Zacpal J.: Concept lattices constrained by equivalence relations. *Proc. CLA 2004*, Ostrava, Czech Republic, pp. 58–66.
3. Bock H. H.: *Automatische Klassifikation.* Vandenhoeck & Ruprecht, Göttingen, 1974.
4. Diognon J.-P., Falmagne J.-C.: *Knowledge Spaces.* Springer, Berlin, 1999.
5. Everitt, Brian S.: *Cluster Analysis, 4th ed.* Edward Arnold, 2001.
6. Ganter B., Wille R.: *Formal Concept Analysis. Mathematical Foundations.* Springer-Verlag, Berlin, 1999.
7. Ganter, Wille: Contextual attribute logic. In: Tepfenhart W., Cyre W. (Eds.): *Proceedings of ICCS 2001*, Springer, 2001.
8. Godin R., Missaoui R., Alaoui H.: Incremental concept formation algorithms based on Galois (concept) lattices. *Computational Intelligence* **11**(2) (1995), 246–267.
9. Kent R. E.: Rough concept analysis. In: Ziarko W. P. (Ed.): *Rough Sets, Fuzzy Sets, and Knowledge Discovery.* Proc. of the Intern. Workshop RSKD'93, Springer-Verlag, London, 1994.
10. Maier D.: *The Theory of Relational Databases.* Computer Science Press, Rockville, 1983.

11. Van der Merwe, F.J., Obiedkov, S., and Kourie, D.G.: AddIntent: A new incremental lattice construction algorithm. In: Concept Lattices. Proc. of the 2nd Int. Conf. on Formal Concept Analysis, Sydney, Australia, Lecture Notes in Artificial Intelligence, vol. 2961, pp. 372–385.
12. Ore O.: Galois connections. *Trans. Amer. Math. Soc.* 55:493–513, 1944.
13. Quinlan J. R.: *C4.5: Programs for Machine Learning.* Morgan-Kaufmann, San Francisco, CA, 1993.
14. Wille R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival I.: *Ordered Sets.* Reidel, Dordrecht, Boston, 1982, 445–470.