Formal Concept Analysis With Background Knowledge: Attribute Priorities

Radim Belohlavek, *Senior Member, IEEE*, and Vilem Vychodil, *Member, IEEE*

***Abstract*—This paper deals with background knowledge in knowledge extraction from binary data. A background knowledge represents an additional piece of information a user may have along with the input data. Such information can be considered as specifying the type of knowledge a user is looking for in the data. In particular, we emphasize the need for taking into account background knowledge in formal concept analysis. We present an approach to modeling background knowledge that represents user's priorities regarding attributes and their relative importance. Such priorities serve as a constraint—only those formal concepts that are compatible with user's priorities are considered relevant, extracted from data, and presented to the user. Our approach has two main practical features. First, the number of formal concepts presented to the user may get significantly reduced. As a result, the user is supplied with relevant formal concepts only and is not overloaded by a large number of possibly nonrelevant formal concepts. Second, different priorities lead to different pieces of knowledge extracted from data. This way, the input data may be repeatedly used in knowledge extraction for different purposes corresponding to different priorities. We concentrate on foundational aspects such as mathematical feasibility, reasoning with background knowledge, removing redundancy from background knowledge, and computational tractability, and present several illustrative examples. In addition, we discuss directions for future research.**

***Index Terms*—Attribute priorities, background knowledge, clustering, formal concept analysis (FCA).**

I. Introduction and Preliminaries

A. Problem Description and Paper Content

BINARY data abound in science, engineering, business, and humanities. Examples include various tabular datasets with rows corresponding to objects, such as products, and columns corresponding to their binary attributes, such as presence of a particular feature. Databases containing descriptions of items, such as customers, in terms of entities to which they are associated, such as services to which they subscribe, or results of questionnaires are further examples of binary data. Recently, formal concept analysis (FCA) proved to be useful for knowledge extraction from and visualization of binary datasets in various application domains such as organization of Web search results into a hierarchical structure of concepts based on common topics [7] (see also http://credo.fub.it/), information retrieval, hierarchical analysis of software code, or visualization in software engineering, to list just a few recent application areas. Further references to applications of FCA can be found in [7] and [12]. A distinguishing feature of FCA is an inherent integration of three components: discovery of clusters (so-called formal concepts) in data, discovery of data dependencies (so-called attribute implications) in data, and visualization of formal concepts and attribute implications by a single hierarchical diagram (so-called concept lattice).

In the basic setting of FCA, the input data consist of a table describing objects and attributes, and their relationship. A background knowledge that a user may have regarding the input data is not taken into account in the basic setting. The background knowledge may reflect additional information regarding the data or a particular type of knowledge that a user is looking for in the data. An absence of background knowledge may result in extraction of a large number of formal concepts including those that may seem artificial, and hence, not interesting to the user because they are not congruent with his background knowledge. On the other hand, an appropriate treatment of background knowledge may result in a focused knowledge extraction. Most importantly, it may reduce the number of formal concept extracted from data. In addition, the information provided by the input data can be used for different purposes that correspond to different contents of background knowledge.

One particular type of background knowledge that we deal with in this paper is relative importance of attributes. Such type of background knowledge is commonly used in human categorization. For instance, when categorizing books for the purpose of inclusion in a sales catalog, one might consider the field subject of a book more important than the type of book. Accordingly, we expect to form categories of books based on the subject, such as "engineering," "computer science," "mathematics," "biology," etc., and only after that, within these categories, we might want to form smaller categories based on the type such as "engineering/textbook," "engineering/research book," etc. In such a case, our background knowledge tells that attributes describing the subject ("engineering," "computer science," etc.) are more important than attributes describing the type of book ("textbook," "research book"). Clearly, the background knowledge depends on the purpose of categorization. For a different purpose, it can be appropriate to use a different background knowledge. For instance, one could

Manuscript received February 23, 2008; revised August 31, 2008 and November 14, 2008; accepted November 26, 2008. This paper was presented in part at the International Conference on Formal Concept Analysis (ICFCA) 2005, the Joint Conference on Information Sciences (JCIS) 2006, and the Association for Computing Machinery (ACM) Symposium on Applied Computing (SAC) 2008. This work was supported by Grant 1ET101370417 of the Grant Agency of the Academy of Sciences (GA AV ČR) and Grant MSM 6198959214. This paper was recommended by Associate Editor.

The authors are with the Department of Systems Science and Industrial Engineering, T. J. Watson School of Engineering and Applied Science, State University of New York (SUNY) at Binghamton, Binghamton, NY 13902-6000 USA and also with the Department of Computer Science, Palacky University, Olomouc 77900, Czech Republic (e-mail: rbelohla@binghamton.edu; vychodil@binghamton.edu).

Digital Object Identifier 10.1109/TSMCC.2008.2012168

consider the type of book more important than the subject. Correspondingly, we would get categories "textbook," "research book," and their subcategories "textbook/engineering," "textbook/computer science," etc. Therefore, while the input data are given (books and their attributes) and do not change, the background knowledge that guides the categorization is purpose-dependent. The relative importance of attributes serves as a constraint in categorization/clustering. Namely, it excludes potential clusters that are not congruent with the background knowledge. For instance, with subject more important than type, category "textbook" consisting of all textbooks (no matter what subject) is not relevant (is not formed in the process of categorization), because it is not congruent with the background knowledge (does not satisfy the corresponding constraint saying that subject is more important than type). Contrary to that, categories "engineering," "engineering/textbook," "computer science," "computer science/textbook" are congruent with the background knowledge. A background knowledge can not only eliminate categories that are not suitable for a given purpose, but also eliminates unnatural categories. As an example, not taking into account that book binding is less important than book subject and book type, one would end up with categories "paperback" and "hardback," which, however logically correct, do not make much sense in a useful categorization of books.

In this paper, we present an approach to the aforementioned type of background knowledge using particular formulas we call attribute-dependency formulas (AD formulas). This approach is theoretically feasible and computationally tractable. Moreover, since the background knowledge is represented by simple logic formulas, it provides us with a user-friendly way to a focused knowledge extraction. We present results on reasoning with background knowledge including entailment and its efficient checking, a syntactico-semantically complete system of deduction rules, and on removing redundancy from background knowledge in a computationally feasible way.

Let us note that the idea of using background knowledge appears in various forms in data mining [6]. Using background knowledge for constraining purposes in clustering has recently been studied in several papers (see, e.g., [8] and [17]). In FCA, a type of background knowledge has been studied in [11] and [16], where the authors use attribute implications, which we discuss later in this paper, for the purpose of attribute exploration. Both the type of the background knowledge and the aims in [11] and [16] are different from those that are discussed in our paper. An approach that is closely related to the one presented in this paper is studied in [2] and [3].

The paper is organized as follows. Section I-B presents preliminaries from FCA. In Section II, we present the approach to and main results on FCA with background knowledge that can be expressed using AD formulas. Section III provides examples, one regarding conceptual classification of felines and one dealing with modeling customer priorities in exploration and visualization of a product catalog. Section IV presents conclusions and directions for future research.

B. Preliminaries

In this section, we summarize basic notions of FCA (see also Section I-A). The reader is referred to [12] for mathematical foundations and [7] for basic introduction, algorithms, and applications. FCA is a method for analysis of object–attribute data tables. An object–attribute data table describing which objects have which attributes can be identified with a triplet $\langle X, Y, I\rangle$, where X is a nonempty set of objects, Y is a nonempty set of attributes, and $I \subseteq X \times Y$ is an object–attribute relation. Objects and attributes correspond to table rows and columns, respectively, and $\langle x, y\rangle \in I$ indicates that object x has attribute y (table entry corresponding to row x and column y contains $\times$ or 1; if $\langle x, y\rangle \notin I$, the table entry contains blank or 0). In terms of FCA, a triplet $\langle X, Y, I\rangle$ is called a *formal context*. For every $A \subseteq X$ and $B \subseteq Y$, denote by $A^{\uparrow}$ a subset of Y and by $B^{\downarrow}$ a subset of X, defined as

$$A^{\uparrow} = \{y \in Y \mid \text{for each } x \in A : \langle x, y\rangle \in I\}$$
$$B^{\downarrow} = \{x \in X \mid \text{for each } y \in B : \langle x, y\rangle \in I\}$$

i.e., $A^{\uparrow}$ is the set of all attributes from Y shared by all objects from A and $B^{\downarrow}$ is the set of all objects from X sharing all attributes from B. A *formal concept* in $\langle X, Y, I\rangle$ is a pair $\langle A, B\rangle$ of $A \subseteq X$ and $B \subseteq Y$ satisfying $A^{\uparrow} = B$ and $B^{\downarrow} = A$. A and B are called the *extent* and *intent* of $\langle A, B\rangle$, respectively, i.e., a formal concept consists of a set A of objects that fall under the concept and a set B of attributes that fall under the concept such that A is the set of all objects sharing all attributes from B and, conversely, B is the collection of all attributes from Y shared by all objects from A. Alternatively, formal concepts can be defined as maximal rectangles of $\langle X, Y, I\rangle$ that are full of $\times$'s (or 1's). For $A \subseteq X$ and $B \subseteq Y$, $\langle A, B\rangle$ is a formal concept in $\langle X, Y, I\rangle$ if and only if $A \times B \subseteq I$ and there is no $A' \supset A$ or $B' \supset B$ such that $A' \times B \subseteq I$ or $A \times B' \subseteq I$. Hence, $\langle A, B\rangle$ is a formal concept if and only if the area corresponding to A (rows) and B (columns) is a maximal rectangular subarea (submatrix) of the input data table that is full of $\times$'s.

The set

$$\mathcal{B}(X, Y, I) = \{\langle A, B\rangle \mid A^{\uparrow} = B, B^{\downarrow} = A\}$$

of all formal concepts in data $\langle X, Y, I\rangle$ can be equipped with a partial order $\leq$ defined by

$$\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle \quad \text{iff } A_1 \subseteq A_2 \quad (\text{iff } B_2 \subseteq B_1). \tag{1}$$

The partial order $\leq$ models the subconcept–superconcept hierarchy. In such hierarchy, for instance, *dog* $\leq$ *mammal*, i.e., the concept *dog* is a subconcept of the concept *mammal*. Note that $^{\uparrow}$ and $^{\downarrow}$ form a Galois connection [12], and that $\mathcal{B}(X, Y, I)$ is, in fact, a set of all fixed points of $^{\uparrow}$ and $^{\downarrow}$. $\mathcal{B}(X, Y, I)$ equipped with $\leq$ happens to be a complete lattice, called the *concept lattice* of $\langle X, Y, I\rangle$. The basic structure of $\mathcal{B}(X, Y, I)$ is described by the so-called basic theorem of concept lattices [12].

Theorem 1: 1) The set $\mathcal{B}(X, Y, I)$ equipped with $\leq$ a complete lattice where the infima and suprema are given by

$$\textstyle\bigwedge_{j\in J}\langle A_j, B_j\rangle = \langle \bigcap_{j\in J} A_j, (\bigcup_{j\in J} B_j)^{\downarrow\uparrow}\rangle$$
$$\textstyle\bigvee_{j\in J}\langle A_j, B_j\rangle = \langle (\bigcup_{j\in J} A_j)^{\uparrow\downarrow}, \bigcap_{j\in J} B_j\rangle.$$

TABLE I
INPUT DATA TABLE

	genus				habitat				size			fur		color			
	Acinonyx	Felis	Leptailurus	Panthera	Africa	America	Asia	Europe	small	medium	large	stripes	spots	black	sandy	white	yellow
Cheetah	×				×					×		×	×				×
Cougar				×		×					×				×		
Jaguar				×		×					×		×	×			×
Lion				×	×						×				×		
Panther				×	×		×			×			×				×
Serval			×		×				×			×	×		×		×
Tiger				×			×				×	×				×	×
Wildcat		×			×	×	×	×	×			×	×	×	×		×

2) Moreover, an arbitrary complete lattice $\mathbf{V} = \langle V, \leq \rangle$ is isomorphic to $\mathcal{B}(X, Y, I)$ iff there are mappings $\gamma : X \to V$, $\mu : Y \to V$ such that the following conditions hold.

a) $\gamma(X)$ is $\bigvee$-dense in V, $\mu(Y)$ is $\bigwedge$-dense in V.

b) $\gamma(x) \leq \mu(y)$ iff $\langle x, y \rangle \in I$.

Note that part 2) of Theorem 1 provides a way to visualize and label concept lattices [12]. For a detailed information on FCA, we refer to [7] and [12], where a reader can find theoretical foundations, methods and algorithms, and applications in various areas.

II. BACKGROUND KNOWLEDGE VIA AD FORMULAS

AD formulas were introduced in [1] and further studied in [4] and [5]. AD formulas serve for representation of relative importance of attributes. A basic motivation for such type of constraints is provided in Section I.

A. AD Formulas and Constrained Concept Lattices

The ideas regarding relative importance of attributes, as described informally before, can be approached in the framework of FCA as follows.

Definition 1: An attribute-dependency formula (AD formula) over a set Y of attributes is an expression

$$A \sqsubseteq B$$

where $A, B \subseteq Y$ are sets of attributes. $A \sqsubseteq B$ *is true in a set* $M \subseteq Y$, written as $M \models A \sqsubseteq B$, if the following condition is satisfied:

$$\text{if } A \cap M \neq \emptyset, \qquad \text{then } B \cap M \neq \emptyset. \tag{2}$$

A formal concept $\langle C, D \rangle \in \mathcal{B}(X, Y, I)$ satisfies $A \sqsubseteq B$ iff $D \models A \sqsubseteq B$. □

Example 1: Table I contains data that we use for illustration in our paper. It describes selected felines (objects) and their characteristics (attributes). Each row of the table is a record of characteristics of a species of felines. The attributes can be divided into four groups: biological genus (*Acinonyx*, *Felis*, *Leptailurus*, and *Panthera*), natural habitat of the species (attributes representing continents), size of its body (small/medium/large), fur pattern (fur with stripes and/or spots), and color in which the species may appear. Thus, the input table consists of eight objects and 17 attributes. A table like this may be constructed by a biologist who wishes to use the information contained in the table to form clusters of species with common properties. This task is, in fact, an application of FCA.

The corresponding concept lattice $\mathcal{B}(X, Y, I)$ contains, among others, the formal concept $\langle C, D \rangle$ with

$$C = \{\text{cheetah, jaguar, panther, serval, tiger, wildcat}\}$$

$$D = \{\text{yellow}\}.$$

This formal concept represents a category/cluster of all felines with yellow fur. Such a category is formed correctly. However, for a biologist, such a category may seem unnatural (there is no such concept as "felines with yellow fur" for a biologist). This is because whenever the biologist considers the color of fur for the purpose of categorization, he always considers other attributes that are more important, such as "being a species that belongs to the *Panthera* genus," i.e., the biologist considers genus more important than color. This background knowledge can be expressed by means of the AD formula

$$\{\text{black, sandy, white, yellow}\} \sqsubseteq \{\textit{Acinonyx}, \textit{Felis}, \textit{Leptailurus}, \textit{Panthera}\}. \tag{3}$$

The previous formal concept $\langle C, D \rangle$ does not satisfy this AD formula because

$$\{\text{black, sandy, white, yellow}\} \cap \{\text{yellow}\} \neq \emptyset$$

but

$$\{\textit{Acinonyx}, \textit{Felis}, \textit{Leptailurus}, \textit{Panthera}\} \cap \{\text{yellow}\} = \emptyset$$

i.e., while attribute "yellow" is used in the description (intent) D of the formal concept $\langle C, D \rangle$, none of the more important attributes specified by AD formula (3) is. On the other hand, formal concept $\langle C, D \rangle$ with

$$C = \{\text{jaguar, panther, tiger}\} \qquad D = \{\textit{Panthera}, \text{yellow}\}$$

satisfies (3). This formal concept corresponds to the category of species within genus *Panthera* that can appear in a yellow color. □

The set of all formal concepts from $\mathcal{B}(X, Y, I)$ that satisfy a given set T of AD formulas is denoted by $\mathcal{B}_T(X, Y, I)$, i.e.

$$\mathcal{B}_T(X, Y, I) = \{\langle C, D \rangle \in \mathcal{B}(X, Y, I) \mid \text{for every } A \sqsubseteq B \in T : D \models A \sqsubseteq B\}$$

and is called the *concept lattice of* $\langle X, Y, I \rangle$ *constrained by* T.

Remark 1: i) In an AD formula $A \sqsubseteq B$, A and B are usually collections of attributes of the same kind such as in (3). This makes it possible to attach an apt meaning to $A \sqsubseteq B$ such as "color is less important than genus."

ii) Equation (2) is just the condition of validity of dependencies considered in the theory of knowledge spaces [9], where M is interpreted as a set of questions an individual can answer and $A \sqsubseteq B$ being true in M means that if the individual fails in answering all questions from A, then he fails in answering all questions from B. However, our aims and results concerning AD formulas are different from those in [9]. □

The following theorem describes the structure of $\mathcal{B}_T(X,Y,I)$.

Theorem 2: $\mathcal{B}_T(X,Y,I)$ is a partially ordered subset of $\mathcal{B}(X,Y,I)$, which is bounded from below. Moreover, if T does not contain an AD formula $A \sqsubseteq B$ such that some $y \in A$ is shared by all objects from X and none attribute from B is shared by all objects from X, then $\mathcal{B}_T(X,Y,I)$ is bounded from above.

Proof: Obviously, $\langle Y^{\downarrow}, Y\rangle$, the least formal concept from $\mathcal{B}(X,Y,I)$, is compatible with every AD formula. Therefore, $\langle Y^{\downarrow}, Y\rangle$ is the lower bound of $\mathcal{B}_T(X,Y,I)$. Furthermore, if there is no AD formula in T with the aforementioned properties, then $\langle X, X^{\uparrow}\rangle$ is the upper bound of $\mathcal{B}_T(X,Y,I)$ since, in this case, $\langle X, X^{\uparrow}\rangle$ clearly satisfies every AD formula from T. ■

Remark 2: Note that the condition guaranteeing that $\mathcal{B}_T(X,Y,I)$ 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. □

The following theorem shows that with a more restricting type of AD formulas, $\mathcal{B}_T(X,Y,I)$ is a complete lattice.

Theorem 3: Let T be a set of AD formulas of the form $y \sqsubseteq y'$. Then, $\mathcal{B}_T(X,Y,I)$ is a complete lattice, which is a $\bigvee$-sublattice of $\mathcal{B}(X,Y,I)$.

Proof: Since $\mathcal{B}_T(X,Y,I)$ is bounded from below (Theorem 2), it suffices to show that $\mathcal{B}_T(X,Y,I)$ is closed under suprema in $\mathcal{B}(X,Y,I)$, i.e., for $\langle A_j, B_j\rangle \in \mathcal{B}_T(X,Y,I)$, we have $\langle (\cap_j B_j)^{\downarrow}, \cap_j B_j\rangle \in \mathcal{B}_T(X,Y,I)$. The latter fact is easy to check. ■

One can show that $\mathcal{B}_T(X,Y,I)$ 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. In such case, the constraints allow us to select only the formal concepts that do not separate certain groups of objects. For instance, the groups may form a partition known from outside.

B. Expressive Power of AD Formulas

AD formulas were introduced in [1] as expressions of the form

$$y \sqsubseteq y_1 \sqcup \cdots \sqcup y_n$$

i.e., $\{y\} \sqsubseteq \{y_1, \ldots, y_n\}$ in the present notation. The present extension to formulas of the form $\{z_1, \ldots, z_m\} \sqsubseteq \{y_1, \ldots, y_n\}$ is not essential. Namely, as can be easily shown, a formal concept $\langle C,D\rangle$ satisfies $\{z_1, \ldots, z_m\} \sqsubseteq \{y_1, \ldots, y_n\}$ if and only if $\langle C,D\rangle$ satisfies every $\{z_i\} \sqsubseteq \{y_1, \ldots, y_n\}$.

An attribute may occur on the left-hand side of several AD formulas from a given set T. For example, we may have two AD formulas, $y \sqsubseteq y_1 \sqcup y_2$ and $y \sqsubseteq y_3 \sqcup y_4$. Then, for a formal concept $\langle A,B\rangle$ to be compatible with both of these formulas, it has to satisfy the following condition: 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 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 such a generalized form. Note that a particular form of this sort 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

$$y \sqsubseteq t(y_1, \ldots, y_n) \tag{4}$$

where $t(y_1, \ldots, y_n)$ is a term over Y defined by: 1) each attribute $y \in Y$ is a term and 2) 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 T' of formulas of the form (4), $\mathcal{B}_{T'}(X,Y,I)$ has an obvious meaning (it consists of all formal concepts from $\mathcal{B}(X,Y,I)$ satisfying all formulas from T'). The following assertion shows that with respect to the possibility of expressing constraints, we do not gain anything new by allowing formulas (4).

Theorem 4: For every set T' of formulas (4), there is a set T of AD formulas such that $\mathcal{B}_{T'}(X,Y,I) = \mathcal{B}_T(X,Y,I)$.

Proof: Let us associate with (4) a formula φ of propositional logic that results by replacing $\sqsubseteq$, $\sqcup$, and $\sqcap$ by propositional connectives of implication, disjunction, and conjunction, respectively. Similarly, let φ_t denote the formula resulting by such replacement from a term t. Given a formal concept $\langle C,D\rangle$, consider the valuation $v : Y \to \{0,1\}$ of propositional symbols defined by $v(y) = 1$ if $y \in D$ and $v(y) = 0$ if $y \notin D$. It is an easy observation that $\langle C,D\rangle$ satisfies $y \sqsubseteq t(y_1, \ldots, y_n)$ if and only if φ is true under the valuation v. It follows from the well-known results on representation of formulas of propositional logic by conjunctive normal forms that there exist terms $t_1, \ldots, t_k$ such that every $y \sqsubseteq t_j$ $(j = 1, \ldots, k)$ is an AD formula and for every valuation v, φ_t has the same truth value as $\varphi_{t_1 \sqcap \cdots \sqcap t_k}$. Hence, $\langle C,D\rangle$ satisfies $y \sqsubseteq t(y_1, \ldots, y_n)$ if and only if $\langle C,D\rangle$ satisfies $y \sqsubseteq t_1 \sqcap \cdots \sqcap t_k$ if and only if $\langle C,D\rangle$ satisfies every AD formula $y \sqsubseteq t_j$ $(j = 1, \ldots, k)$. Therefore, if we denote by T' the set of AD formulas that results from T by replacing every $y \sqsubseteq t(y_1, \ldots, y_n)$ by the AD formulas $y \sqsubseteq t_j$ $(j = 1, \ldots, k)$, we get $\mathcal{B}_{T'}(X,Y,I) = \mathcal{B}_T(X,Y,I)$, completing the proof. ■

A particularly interesting structure of clusters in data is that of a tree (see [10]). Trees can be extracted from concept lattices by AD formulas in a natural way. We present a criterion under which $\mathcal{B}_T(X,Y,I)$ is a tree. Since $\mathcal{B}_T(X,Y,I)$ always has the least element (see Theorem 1), we call $\mathcal{B}_T(X,Y,I)$ a tree if $\mathcal{B}_T(X,Y,I) - \{\langle Y^{\downarrow}, Y\rangle\}$ is a tree. Note that $\mathcal{B}_T(X,Y,I) - \{\langle Y^{\downarrow}, Y\rangle\}$ results by deleting the least formal concept, i.e., $\{\langle Y^{\downarrow}, Y\rangle\}$, from $\mathcal{B}_T(X,Y,I)$.

Theorem 5: Let T be a set of AD formulas. If for each $y_1, y_2 \in Y$ that are not disjoint (i.e., $\{y_1\}^{\downarrow} \cap \{y_2\}^{\downarrow} \neq \emptyset$), T 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 of the formulas are pairwise disjoint, then $\mathcal{B}_T(X,Y,I)$ is a tree.

Proof: If $\mathcal{B}_T(X,Y,I)$ is not a tree, there are distinct and noncomparable $\langle A_1, B_1\rangle$, $\langle A_2, B_2\rangle \in \mathcal{B}_T(X,Y,I)$ such that $A_1 \cap A_2 \neq Y^{\downarrow}$, i.e., the extent of the infimum of $\langle A_1, B_1\rangle$ and

$\langle A_2, B_2\rangle$ is greater than the extent of 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, T contains an AD formula $y_1 \sqsubseteq \cdots \sqcup y_2 \sqcup \cdots$, or $y_2 \sqsubseteq \cdots \sqcup y_1 \sqcup \cdots$, case that can be handled symmetrically. 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' that appears on the right-hand side of the AD formula $y_1 \sqsubseteq \cdots \sqcup y_2 \sqcup \cdots$. This yields $\langle x, y'\rangle \in I$. Putting together, we have $\langle x, y_2\rangle \in I$ and $\langle x, y'\rangle \in I$, which contradicts the disjointness of y_2 and y'. ■

Remark 3: The assumption of Theorem 5 is satisfied in the following situation, which often occurs: 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, green, blue, ... Then, we linearly order Y_i's, e.g., by $Y_1 < \cdots < Y_n$, 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 wants to see the formal concepts respecting this order. □

Remark 4: In fact, the 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 T. It is sufficient if they are entailed by T (see Section II-C for the concept of entailment). □

C. Entailment and its Efficient Checking

A background knowledge specified by means of AD formulas may be redundant. For instance, suppose the background knowledge consists of (3) and AD formulas

$$\{\text{black}, \text{sandy}, \text{white}, \text{yellow}\} \sqsubseteq \{\text{Africa}, \text{America}, \text{Asia}, \text{Europe}\} \quad (5)$$

$$\{\text{Africa}, \text{America}, \text{Asia}, \text{Europe}\} \sqsubseteq \{\textit{Acinonyx}, \textit{Felis}, \textit{Leptailurus}, \textit{Panthera}\} \quad (6)$$

i.e., AD formulas (3), (5), and (6) say "color is less important than genus," "color is less important than habitat," and "habitat is less important than genus," respectively. Intuitively, (3) is redundant because it is entailed by (5) and (6). One may wish to remove such redundancy because it makes the description of background knowledge less comprehensible. Shortly, we make the notion of entailment precise and present results that lead to an efficient algorithm for testing entailment and removal of redundancy.

Definition 2: A set $M \subseteq Y$ is called a *model* of a set T of AD formulas if for each $A \sqsubseteq B \in T$, $M \models A \sqsubseteq B$. Let $\mathrm{Mod}(T)$ denote the set of all models of T, i.e.

$$\mathrm{Mod}(T) = \{M \subseteq Y \mid \text{for each } A \sqsubseteq B \in T:\ M \models A \sqsubseteq B\}.$$

$A \sqsubseteq B$ *semantically follows from* T (T *semantically entails* $A \sqsubseteq B$) if for each $M \in \mathrm{Mod}(T)$, we have $M \models A \sqsubseteq B$. □

The following theorem, known from knowledge spaces [9], describes the structure of $\mathrm{Mod}(T)$.

Theorem 6: Let T be a set of AD formulas. Then, $\mathrm{Mod}(T)$ is an interior system, i.e., $\mathrm{Mod}(T)$ is closed under arbitrary unions.

By virtue of Theorem 6, for each set T of AD formulas, we can consider the associated interior operator $I_T : 2^Y \to 2^Y$ defined by $I_T(M) = \bigcup\{N \in \mathrm{Mod}(T) \mid N \subseteq M\}$. Clearly, $\mathrm{Mod}(T) = \{M \subseteq Y \mid M = I_T(M)\}$. Furthermore, for every $M \subseteq Y$, $I_T(M)$ is the greatest model of T that is included in M. The following algorithm shows a way to compute $I_T(M)$, given T and M.

Algorithm 1.

Input: set T of AD-formulas, $M \subseteq Y$
Output: $I_T(M)$

set N ***to*** M
while there is $A \sqsubseteq B \in T$ ***such that*** $N \not\models A \sqsubseteq B$:
  choose $A \sqsubseteq B \in T$ ***and*** $y \in A$ ***such that*** $y \in N$ ***and*** $B \cap N = \emptyset$
  remove y ***from*** N
return N

The next theorem shows a crucial property. Namely, testing entailment can be performed by checking validity in a single model: T entails $A \sqsubseteq B$ iff $A \sqsubseteq B$ is true in the greatest model of T that is contained in the complement $\overline{B}$ of B. Note that testing entailment using a single model is known in other areas too (see, e.g., [12] for attribute implications or [14] for logic programming).

Theorem 7: Let T be a set of AD formulas, $A \sqsubseteq B$ be an AD formula. The following conditions are equivalent.

i) $T \models A \sqsubseteq B$;
ii) $I_T(\overline{B}) \models A \sqsubseteq B$;
iii) $A \cap I_T(\overline{B}) = \emptyset$.

Proof: "i) ⇒ ii)": $T \models A \sqsubseteq B$ means that $A \sqsubseteq B$ is true in each model of T. Since $I_T(\overline{B})$ is a model of T, we immediately get ii).

"ii) ⇒ iii)": By definition, $I_T(\overline{B}) \models A \sqsubseteq B$ iff $A \cap I_T(\overline{B}) \neq \emptyset$ implies $B \cap I_T(\overline{B}) \neq \emptyset$. The latter is true iff $B \cap I_T(\overline{B}) = \emptyset$ implies $A \cap I_T(\overline{B}) = \emptyset$. Now, since $I_T(\overline{B}) \subseteq \overline{B}$, we get $B \cap I_T(\overline{B}) = \emptyset$, which yields $A \cap I_T(\overline{B}) = \emptyset$.

"iii) ⇒ i)": Let $A \cap I_T(\overline{B}) = \emptyset$. It suffices to check that for each model $M \in \mathrm{Mod}(T)$, $M \models A \sqsubseteq B$. We show that under the assumption of $B \cap M = \emptyset$, we get $A \cap M = \emptyset$. Thus, let $B \cap M = \emptyset$. This yields $M \subseteq \overline{B}$. Moreover, we get $M = I_T(M) \subseteq I_T(\overline{B})$ because I_T is monotone, and M is closed under I_T (recall that M is a model of T). Now, since we assume $A \cap I_T(\overline{B}) = \emptyset$, the latter observation $M \subseteq I_T(\overline{B})$ yields $A \cap M = \emptyset$, which proves the claim. ■

Thus, Algorithm 1 and Theorem 7 give us the following algorithm.

Algorithm 2.

Input: set T of AD-formulas and AD-formula $A \sqsubseteq B$
Output: ***true*** if $T \models A \sqsubseteq B$, ***false*** otherwise.
compute $I_T(\overline{B})$ ***using Algorithm 1***
if $A \cap I_T(\overline{B}) = \emptyset$:
  return true
else:
  return false

D. Complete System of Deduction Rules

The next issue related to entailment and reasoning with AD formulas is the question of whether there is a complete system of deduction rules for reasoning with AD formulas. We present a positive answer by showing a system of Armstrong-like rules. We use deduction rules of the form

$$\frac{A_1 \sqsubseteq B_1, \ldots, A_n \sqsubseteq B_n}{A \sqsubseteq B}. \tag{7}$$

These rules are to be used in proofs in a standard way, i.e., having AD formulas that match the "input part" of the rule, i.e., $A_1 \sqsubseteq B_1, \ldots, A_n \sqsubseteq B_n$, we can infer, in a single step, the AD formula corresponding to the "output part," i.e., $A \sqsubseteq B$. In particular, we use the following system of deduction rules:

$$(\overline{\text{Ref}}) \quad \overline{A \sqsubseteq A}$$

$$(\overline{\text{Wea}}) \quad \frac{A \sqsubseteq B}{A \sqsubseteq B \cup C}$$

$$(\overline{\text{Cut}}) \quad \frac{A \sqsubseteq B,\ C \sqsubseteq A \cup D}{C \sqsubseteq B \cup D}$$

for each $A, B, C, D \subseteq Y$. The notions of a proof and provability are defined the usual way, i.e., a *proof* of an AD formula $A \sqsubseteq B$ from a set T of AD formulas is a sequence $\varphi_1, \ldots, \varphi_n$ of AD formulas such that $\varphi_n = A \sqsubseteq B$ and each φ_i either is from T or can be inferred from some of the preceding formulas φ_j, $j < i$, using some of the deduction rules $(\overline{\text{Ref}})$–$(\overline{\text{Cut}})$. An AD formula $A \sqsubseteq B$ is *provable* from a set T of AD formulas if there is a proof of $A \sqsubseteq B$ from T. In such case, we write

$$T \vdash A \sqsubseteq B.$$

Before we turn to a completeness theorem for reasoning with AD formulas, let us observe a close connection between AD formulas and attribute implications. Recall [12] that an attribute implication is an expression $A \Rightarrow B$, where $A, B \subseteq Y$ are sets of attributes. $A \Rightarrow B$ is true in a set $M \subseteq Y$ iff $A \subseteq M$ implies $B \subseteq M$. This fact is denoted by $M \models A \Rightarrow B$. Note that if M is a set of attributes of some object, then $M \models A \Rightarrow B$ means that if the object has all attributes from A, it also has all attributes from B. The basic connection between AD formulas and attribute implications is described by the following lemma whose proof follows directly from definitions.

Lemma 1: For $A, B, M \subseteq Y$, we have $M \models A \sqsubseteq B$ iff $\overline{M} \models B \Rightarrow A$, where $\overline{M} = Y - M$. ■

The following assertion is a completeness theorem for reasoning with AD formulas.

Theorem 8 (Completeness): $(\overline{\text{Ref}})$–$(\overline{\text{Cut}})$ is a sound and complete system of deduction rules, i.e., for any set T of AD formulas and an AD formula $A \sqsubseteq B$, we have

$$T \vdash A \sqsubseteq B \qquad \text{if and only if } T \models A \sqsubseteq B.$$

In words, $A \sqsubseteq B$ is entailed by T iff $A \sqsubseteq B$ can be obtained from T by rules $(\overline{\text{Ref}})$–$(\overline{\text{Cut}})$.

Proof of Theorem 8: Let $T \vdash A \sqsubseteq B$ and let $\varphi_1, \ldots, \varphi_n$ be a proof of $A \sqsubseteq B$. It is easy to see that a sequence $\varphi_1, \ldots, \varphi_n$ of AD formulas is a proof from T using rules (R), ... iff the corresponding sequence $(\varphi_1)^{\text{AI}}, \ldots, (\varphi_n)^{\text{AI}}$ of attribute implications is a proof from T^{AI} using rules $(\text{R})^{\text{AI}}$, ... Here, $(\cdots)^{\text{AI}}$ denotes the result of replacing all AD formulas in $(\cdots)$ by the corresponding attribute implications [12]. For example, $(A \sqsubseteq B)^{\text{AI}} = B \Rightarrow A$, $(\overline{\text{Ref}})^{\text{AI}}$, $(\overline{\text{Wea}})^{\text{AI}}$, and $(\overline{\text{Cut}})^{\text{AI}}$ become

$$\overline{A \Rightarrow A} \qquad \frac{B \Rightarrow A}{B \cup C \Rightarrow A} \qquad \frac{B \Rightarrow A,\ A \cup D \Rightarrow C}{B \cup D \Rightarrow C}$$

and so on. Now, $(\overline{\text{Ref}})^{\text{AI}}$, $(\overline{\text{Wea}})^{\text{AI}}$, and $(\overline{\text{Cut}})^{\text{AI}}$ are the well-known Armstrong rules of reflexivity, weakening, and cut, respectively (see [15]), which are known to be complete w.r.t. semantics of attribute implications (alternatively, one can use the semantics of functional dependencies) (see [12] and [15]). Therefore, $(\varphi_1)^{\text{AI}}, \ldots, (\varphi_n)^{\text{AI}}$ is a proof of $(A \sqsubseteq B)^{\text{AI}}$ from T^{AI} using $(\overline{\text{Ref}})^{\text{AI}}$, $(\overline{\text{Wea}})^{\text{AI}}$, and $(\overline{\text{Cut}})^{\text{AI}}$. Hence, due to the aforementioned completeness, $T^{\text{AI}} \models (A \sqsubseteq B)^{\text{AI}}$. Now, using Lemma 1, one can see that $T^{\text{AI}} \models (A \sqsubseteq B)^{\text{AI}}$ is equivalent to $T \models A \sqsubseteq B$. ■

Note that due to the relationships to attribute implications, we can automatically retrieve further sound deduction rules over AD formulas from the well-known rules for attribute implications (or, equivalently, functional dependencies) such as

$$(\overline{\text{Add}}) \quad \frac{B \sqsubseteq A,\ C \sqsubseteq A}{B \cup C \sqsubseteq A}$$

$$(\overline{\text{Pro}}) \quad \frac{A \cup B \sqsubseteq C}{A \sqsubseteq C}$$

$$(\overline{\text{Tra}}) \quad \frac{A \sqsubseteq B,\ B \sqsubseteq C}{A \sqsubseteq C}.$$

E. Nonredundant Bases of AD Formulas

We now consider the problem of removing redundancy from a given set of AD formulas. We present an algorithm that computes a so-called nonredundant base for a given set of AD formulas. In the standard database terminology, nonredundant bases are known as nonredundant covers [15]. Put verbally, a nonredundant base of a set T of AD formulas is a minimal set T' of AD formulas that entails the same AD formulas as T. Thus, a nonredundant base describes the same information (via semantic entailment) as the original set T of AD formulas, and moreover, provides us with the most economic description of T because it is minimal. This has particular relevance in data analysis because smaller sets of formulas are easier to handle and more comprehensible for a user. Moreover, it decreases

computational costs. For instance, Algorithm 2 runs faster with smaller T's.

We first introduce the notion of an equivalence of sets of AD formulas and show its basic properties. Given sets T_1, T_2 of AD formulas, we say that T_1 *and* T_2 *are equivalent*, written $T_1 \equiv T_2$, if for each $\varphi \in T_1$ and $\psi \in T_2$, we have $T_1 \models \psi$ and $T_2 \models \varphi$.

Theorem 9: Let T_1, T_2 be sets of AD formulas. Then the following are equivalent:

i) $\mathrm{Mod}(T_1) = \mathrm{Mod}(T_2)$;
ii) for each AD formula φ, $T_1 \models \varphi$ iff $T_2 \models \varphi$;
iii) $T_1 \equiv T_2$.

Proof: "i) $\Rightarrow$ ii)": Follows by definition of "$\models$".

"ii) $\Rightarrow$ iii)": Take $\varphi \in T_1$. We have $T_1 \models \varphi$ because φ is true in each model of T_1. Now, ii) gives $T_2 \models \varphi$. Dually, $\psi \in T_2$ gives $T_1 \models \psi$. Hence, $T_1 \equiv T_2$.

"iii) $\Rightarrow$ i)": Let $T_1 \equiv T_2$. By contradiction, let us assume that $\mathrm{Mod}(T_1) \neq \mathrm{Mod}(T_2)$. Thus, either $\mathrm{Mod}(T_1) \not\subseteq \mathrm{Mod}(T_2)$ or $\mathrm{Mod}(T_2) \not\subseteq \mathrm{Mod}(T_1)$. Suppose $\mathrm{Mod}(T_1) \not\subseteq \mathrm{Mod}(T_2)$. Then, there is $M \in \mathrm{Mod}(T_1)$ such that $M \notin \mathrm{Mod}(T_2)$. This means that there is $A \subseteq B \in T_2$ such that $M \not\models A \subseteq B$. On the other hand, $T_1 \equiv T_2$ yields $T_1 \models A \subseteq B$, i.e., $M \models A \subseteq B$ because M is a model of T_1, a contradiction. Dually, one can proceed for $\mathrm{Mod}(T_2) \not\subseteq \mathrm{Mod}(T_1)$. ∎

Coming back to our motivation, for a given T, we want to find the least T' such that $T \equiv T'$.

Definition 3: A set T' of AD formulas is called a nonredundant base of T if $T \equiv T'$ and there is no $T'' \subset T'$ with $T'' \equiv T$. A set T' of AD formulas is called a minimal base of T if $T \equiv T'$ and for each T'' such that $T \equiv T''$, we have $|T'| \leq |T''|$. □

Obviously, if T' is a minimal base of T, then T' is a nonredundant base of T (but not *vice versa* in general).

We are interested in computing nonredundant (or minimal) bases of given sets of AD formulas. A naive way to find a nonredundant base of T is to start with T and remove AD formulas from T until no AD formula from T follows from the others. Such a procedure, however, does not ensure that a nonredundant base found this way is minimal. In what follows we present an algorithm that generates a minimal base of any T. We use an indirect procedure and make use of the following theorem.

Theorem 10: Let $I : 2^Y \to 2^Y$ be an interior operator. Then there is a set T of AD formulas such that $I = I_T$.

Proof: Take an interior operator $I : 2^Y \to 2^Y$. Let T be a set of AD formulas defined by

$$T = \{\overline{I(M)} \subseteq \overline{M} \,|\, M \subseteq Y\}. \tag{8}$$

We prove that $M \subseteq Y$ is a model of T iff $M = I(M)$. First, we show that each fixed point of I is a model of T. Thus, consider $N \subseteq Y$ such that $N = I(N)$ and take an arbitrary $\overline{I(M)} \subseteq \overline{M} \in T$. In order to prove $N \models \overline{I(M)} \subseteq \overline{M}$, it suffices to show that $N \cap \overline{M} = \emptyset$ implies $N \cap \overline{I(M)} = \emptyset$. From $N \cap \overline{M} = \emptyset$, it follows that $N \subseteq M$. Furthermore, monotony of the interior operator I yields $N = I(N) \subseteq I(M)$, i.e., $N \cap \overline{I(M)} = \emptyset$, which proves that each fixed point of I is a model of T.

Now, let $M \neq I(M)$, i.e., $I(M) \subset M$, and consider $\overline{I(M)} \subseteq \overline{M} \in T$. Observe that $M \cap \overline{M} = \emptyset$. In addition to that, $\overline{I(M)} \cap M \neq \emptyset$ because $I(M) \subset M$. Thus, $M \not\models \overline{I(M)} \subseteq \overline{M}$. As a consequence, $M \notin \mathrm{Mod}(T)$, finishing the proof. ∎

We proceed as follows. We associate with any T a closure operator C (see, e.g., [12]). We then use Ganter's NEXTCLOSURE algorithm [12] to compute a minimal base (of attribute implications) associated to C and determine a minimal base of T. Note that the minimal base associated to C, which we use, is the well-known Guigues–Duquenne base [13]. The resulting algorithm allows us to list a minimal base of T with a polynomial time delay. The algorithm is sketched below.

Algorithm 3.

Input: set T of AD-formulas
Output: minimal base $T_{\min}$ of T

define $C\colon 2^Y \to 2^Y$ ***by*** $C(M) = \overline{I_T(\overline{M})}$
set P ***to*** $\emptyset$
while $P \neq Y$:
  if $P \neq C(P)$:
    add $P \Rightarrow C(P)$ ***to*** T'
  set P ***to*** NEXTCLOSURE(P, T')
set $T_{\min} = \{B - A \subseteq A \,|\, A \Rightarrow B \in T'\}$
return $T_{\min}$

Example 2: Consider the following set of AD formulas:

$$T = \{a \subseteq \{b,c,e\}, b \subseteq \{c,d,e\}, b \subseteq \{b,c,d\},$$
$$c \subseteq \{a,b,d\}, d \subseteq \{c,e\}, d \subseteq \{b,d\}, e \subseteq \{c,d\}\}.$$

The minimal base $T_{\min}$ of T that is computed by Algorithm 3 is the following:

$$\{\{c,e\} \subseteq \{a,b,d\}, \{a,b,e\} \subseteq \{c,d\}, \{a,b,d\} \subseteq \{c,e\}\}.$$

Hence, compared to the original set T that consists of seven AD formulas, $T_{\min}$ consists of three AD formulas that encompass the same information about attribute dependencies as T. □

III. EXAMPLES

A. Conceptual Categorization of Felines

Consider again the data from Table I (cf., Example 1). The corresponding concept lattice $\mathcal{B}(X, Y, I)$ contains 35 conceptual clusters (formal concepts) and is depicted in Fig. 1. As shown in Example 1, if we do not include any background knowledge, we obtain all formal concepts including nonnatural ones, such as the formal concept corresponding to the category of felines with yellow fur. Suppose we impose constraints by adding background knowledge using the following AD formulas:

$$\{\text{stripes}, \text{spots}, \text{black}, \text{sandy}, \text{white}, \text{yellow}\} \subseteq \{\text{small}, \text{medium}, \text{large}\} \tag{9}$$

$$\{\text{small}, \text{medium}, \text{large}\} \subseteq \{\textit{Acinonyx}, \textit{Felis}, \textit{Leptailurus}, \textit{Panthera}\} \tag{10}$$

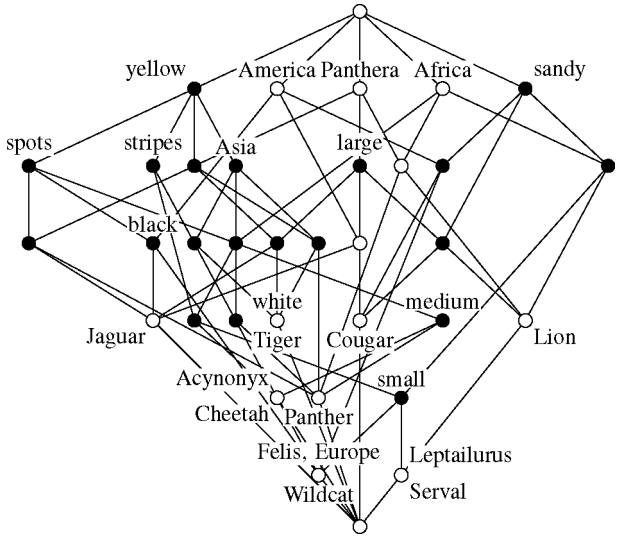

Fig. 1. Hierarchy of all formal concepts of data from Table I. The formal concepts are represented by nodes and are labeled by objects and attributes in a standard way [12], i.e., a node representing a formal concept $\langle A, B\rangle$ is labeled by object x if $\langle A, B\rangle$ is the smallest formal concept that contains x; a node representing a formal concept $\langle A, B\rangle$ is labeled by attribute y if $\langle A, B\rangle$ is the largest formal concept that contains y. Both the extent A and the intent B of a formal concept $\langle A, B\rangle$ represented by a node n can be retrieved from the diagram. Namely, A contains object x if and only if there is a path going downward from n to a node labeled by x; B contains attribute y if and only if there is a path going upward from n to a node labeled by y.

$$\{\text{small}, \text{medium}, \text{large}\} \sqsubseteq \{\text{Africa}, \text{America}, \text{Asia}, \text{Europe}\}. \tag{11}$$

These formulas imply that fur color and pattern are less important than size, size is less important than genus, and size is less important than habitat, respectively. No priority is asserted between habitat and genus, nor between fur color and pattern. The corresponding constrained concept lattice $\mathcal{B}_T(X, Y, I)$ consists of 15 conceptual clusters (the names of attributes are abbreviated)

$$C_1 = \langle X, \emptyset\rangle$$
$$C_2 = \langle\{\text{cougar}, \text{jaguar}, \text{wildcat}\}, \{\text{Am}\}\rangle$$
$$C_3 = \langle\{\text{cheetah}, \text{lion}, \text{panther}, \text{serval}, \text{wildcat}\}, \{\text{Af}\}\rangle$$
$$C_4 = \langle\{\text{cougar}, \text{jaguar}, \text{lion}, \text{panther}, \text{tiger}\}, \{\text{Pa}\}\rangle$$
$$C_5 = \langle\{\text{tiger}\}, \{\text{Pa}, \text{As}, \text{la}, \text{st}, \text{wh}, \text{ye}\}\rangle$$
$$C_6 = \langle\{\text{cougar}, \text{jaguar}\}, \{\text{Pa}, \text{Am}, \text{la}\}\rangle$$
$$C_7 = \langle\{\text{cougar}\}, \{\text{Pa}, \text{Am}, \text{la}, \text{sa}\}\rangle$$
$$C_8 = \langle\{\text{jaguar}\}, \{\text{Pa}, \text{Am}, \text{la}, \text{sp}, \text{bl}, \text{ye}\}\rangle$$
$$C_9 = \langle\{\text{lion}, \text{panther}\}, \{\text{Pa}, \text{Af}\}\rangle$$
$$C_{10} = \langle\{\text{lion}\}, \{\text{Pa}, \text{Af}, \text{la}, \text{sa}\}\rangle$$
$$C_{11} = \langle\{\text{panther}\}, \{\text{Pa}, \text{Af}, \text{As}, \text{me}, \text{sp}, \text{ye}\}\rangle$$
$$C_{12} = \langle\{\text{serval}\}, \{\text{Le}, \text{Af}, \text{sm}, \text{st}, \text{sp}, \text{sa}, \text{ye}\}\rangle;$$
$$C_{13} = \langle\{\text{wildcat}\}, \{\text{Fe}, \text{Af}, \text{Am}, \text{As}, \text{Eu}, \text{sm}, \text{st}, \text{sp}, \text{bl}, \text{sa}, \text{ye}\}\rangle$$

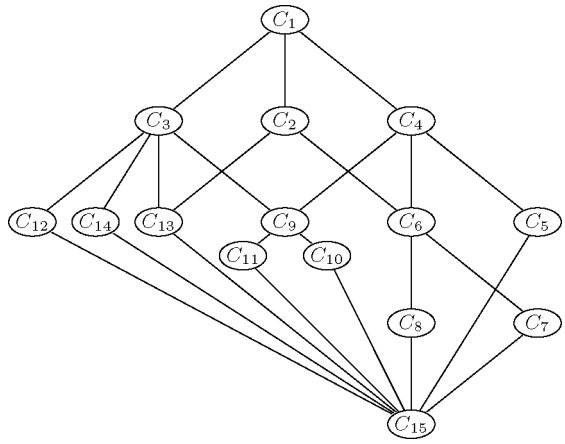

Fig. 2. Concept lattice of data from Table I constrained by (9)–(11).

$$C_{14} = \langle\{\text{cheetah}\}, \{\text{Ac}, \text{Af}, \text{me}, \text{st}, \text{sp}, \text{ye}\}\rangle$$
$$C_{15} = \langle\emptyset, Y\rangle.$$

The hierarchy of these clusters is depicted in Fig. 2. As one can see, the new hierarchy is considerably easier to comprehend than the original one and does not contain "artificial" formal concepts, such as the one corresponding to the category of all felines with yellow fur. Returning to Fig. 1, the black nodes represent clusters that are omitted in Fig. 2 while the white ones represent clusters that are present in Fig. 2. The hierarchy in Fig. 2 contains two clusters that are trivial: C_1 (cluster of all animals) and C_{15} (cluster of no animals). These two borderline clusters may be omitted in the diagram.

One of the benefits of adding background knowledge to reduce the concept lattice is its interactive character. Namely, if a user supplies AD formulas and the structure is still too large, he can specify further restrictions that may help to further reduce the structure. For illustration, suppose we add the following AD formula to the previous ones:

$$\{\text{Af}, \text{Am}, \text{As}, \text{Eu}\} \sqsubseteq \{\text{bl}, \text{sa}, \text{wh}, \text{ye}\}. \tag{12}$$

The AD formula says that if a specification of a continent is present, then the specification of color must also be present in the concept description (intent). Using this AD formula as an additional constraint, we arrive at 11 formal concepts as clusters. Each object (table row) induces one formal concept—the category of the corresponding species. In addition, there is a formal concept C_4 (category of all *Panthera* felines) that appears in the data and is compatible with the background knowledge. The resulting constrained concept lattice is depicted in Fig. 3. From Fig. 3, we can see that some formal concepts such as C_6 (category of all large *Panthera* felines from America) are no longer present in the hierarchy because they do not satisfy the constraint represented by (12).

B. Customer Priorities in Product Catalogs

In this section, we demonstrate how AD formulas can be used to represent customer priorities on attributes of products and how such priorities can enhance visualization and exploration

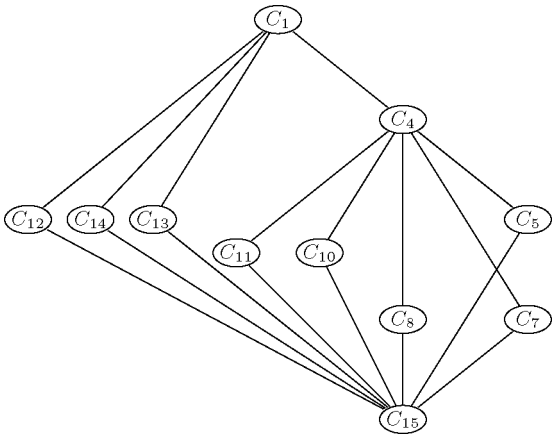

Fig. 3. Concept lattice of data from Table I constrained by (9)–(11) and (12).

TABLE II
INPUT DATA TABLE: WORKSTATIONS AND THEIR FEATURES

Model	Arch.		Mnfctr.			CPUs		Cores			Clock			RAM			HDD			Price		
	86	sp	A	I	S	1	2	1	2	4	1	m	h	8	16	32	1	2	3	1k	2k	3k
20	×		×			×		×	×			×		×			×			×		
24	×			×		×			×	×			×	×					×	×		
40	×		×			×	×	×	×				×	×	×	×		×			×	
25		×			×	×		×			×			×			×				×	
45		×			×	×	×	×				×		×	×		×					×

of products by means of concept lattices. We demonstrate that by using AD formulas, users can reduce the number of groups of products (product clusters) extracted by ordinary FCA by focusing on relevant groups, i.e., groups that are congruent with the priorities. This way, our method improves user's ability to search, evaluate, and compare different products by means of their attributes.

We consider an example where products are computer workstations supplied in certain configurations. The attributes are particular parameters of the configurations like the number of CPUs, CPU speed, maximum memory available, etc. Having several configurations available, a user may wish to choose among the configurations based on his priorities. Some users may prefer total mass storage capacity over CPU speed (e.g., users interested in database applications), other users may wish to have a system with faster CPU and larger memory (e.g., for computer-aided design (CAD) applications), etc. Constraints like these are naturally expressed through AD formulas. In our example, we use Sun Ultra Workstations, manufactured by Sun Microsystems, which come in several different configurations. The data were taken from www.sun.com (all models are from November 2007). Using information from Sun's Web page, we have constructed a data table describing important parameters of the workstations [see Table II (left)].

The rows of the table (objects) represent models of Sun Ultra Workstations: 20, 24, 40, 25, and 45. We consider 22 attributes divided into eight groups, as described in Table II (right).

The concept lattice $\mathcal{B}(X, Y, I)$ associated to Table II (left) contains 17 formal concepts. The concepts are shown in Table III. For instance, the extent of formal concept no. 4 contains models 25 and 45, and its intent contains attributes sp (architecture: UltraSparc), S (manufacturer: Sun), 1 (one CPU), etc. The corresponding concept lattice $\mathcal{B}(X, Y, I)$ is depicted in Table II (right).

TABLE III
ALL FORMAL CONCEPTS IN DATA

	Extent					Intent																									
	Model					Arch.		Mnfctr.			CPUs		Cores			Clock			RAM			HDD			Price		
	20	24	40	25	45	86	sp	A	I	S	1	2	1	2	4	1	m	h	8	16	32	1	2	3	1k	2k	3k
1						×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
2					×		×			×	×	×	×				×		×	×		×					×
3				×			×			×	×		×			×			×			×				×	
4				×	×		×			×	×		×						×			×					
5			×			×		×			×	×	×	×				×	×	×	×		×			×	
6			×		×						×	×	×						×	×							
7			×	×							×		×						×							×	
8		×				×			×		×			×	×			×	×						×	×	
9		×	×			×					×			×				×	×								
10	×					×		×			×		×	×			×		×			×			×		
11	×				×						×		×				×		×			×					
12	×			×	×						×								×			×					
13	×		×			×		×			×		×	×					×								
14	×		×	×	×						×		×						×								
15	×	×				×					×			×					×						×		
16	×	×	×			×					×			×					×								
17	×	×	×	×	×						×								×								

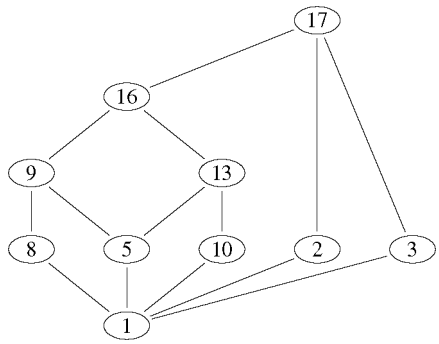

Fig. 4. Concept lattice of data from Table II (left) constrained by (13)–(15).

Suppose that a customer is interested in finding workstations according to the following priorities.

1) Mass storage capacity is more important than price.
2) CPU clock speed is more important than mass storage capacity.
3) Processor architecture is more important than number of processor cores.

Priorities of this type may be imposed by, e.g., a customer interested in high-speed graphics workstation. In terms of AD formulas, we represent this particular constraint as follows:

$$\{1k, 2k, 3k\} \sqsubseteq \{\text{HDD1}, \text{HDD2}, \text{HDD3}\} \tag{13}$$

$$\{\text{HDD1}, \text{HDD2}, \text{HDD3}\} \sqsubseteq \{l, m, h\} \tag{14}$$

$$\{\text{Cores1}, \text{Cores2}, \text{Cores4}\} \sqsubseteq \{86, sp\}. \tag{15}$$

Of all the original concepts, ten satisfy such constraint: 1, 2, 3, 5, 8, 9, 10, 13, 16, and 17. For instance, formal concept no. 4 does not satisfy the constraint because it specifies mass storage information but not the CPU clock speed. Such formal concept is not interesting for our user, who prefers the information about CPU speed over the one about the mass storage capacity. The constrained concept lattice $\mathcal{B}_T(X, Y, I)$ corresponding to the previous set T of AD formulas is shown in Fig. 4.

Often, users feel that certain attributes are equally important. For instance, CPU manufacturer and CPU clock speed can both be important and a user may wish to have information about both

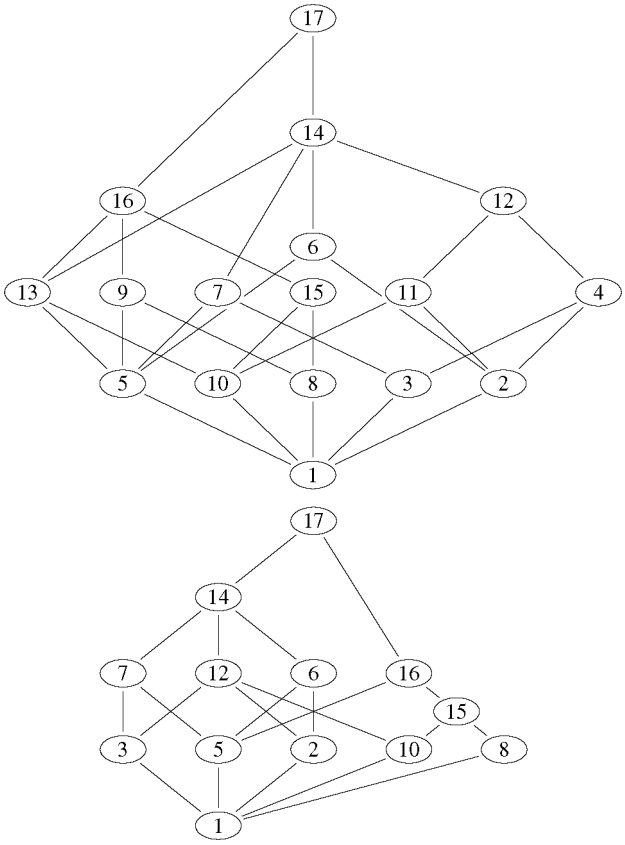

Fig. 5. Concept lattice of data from Table II (left) constrained by (16)–(17).

of them or, on a more general level in the conceptual hierarchy, none of them at the same time. From the point of view of AD formulas, we can formulate such a constraint as follows:

$$\{A, I, S\} \sqsubseteq \{l, m, h\} \tag{16}$$

$$\{l, m, h\} \sqsubseteq \{A, I, S\}. \tag{17}$$

The AD formulas say that whenever the information about the architecture is present, the information about the speed must also be present and *vice versa*. If we use this constraint, we obtain 13 concepts. The formal concepts that do not satisfy the constraint are 4, 9, 11, and 13. The constrained concept lattice $\mathcal{B}_U(X, Y, I)$ corresponding to the previous set U of AD formulas is shown in Fig. 5.

C. Removing Redundancy

As mentioned in Section II-E, removing redundant AD formulas can improve the readability of background knowledge. Background knowledge represented by AD formulas may be collected from different sources, by different people, and during a longer period of time. In addition, if we take into account the number of attributes, which can be large, it is likely that the "same information" will be captured by different groups of rules in the knowledge base. The redundancy thus involved is undesirable because it impairs comprehensibility of the background knowledge. We now demonstrate how Algorithm 2 can be used to remove redundant AD formulas. As an example, consider a background knowledge represented by the following AD formulas over attributes $a, \ldots, g$:

$$\{a, b\} \sqsubseteq \{c\} \qquad \{c, d\} \sqsubseteq \{a\} \qquad \{c, e\} \sqsubseteq \{a, d\}$$
$$\{f, g\} \sqsubseteq \{a, c\} \qquad \{d, f\} \sqsubseteq \{b, c\} \qquad \{g\} \sqsubseteq \{e, f\}.$$

The AD formula $\{f, g\} \sqsubseteq \{a, c\}$ is redundant. In other words, $\{f, g\} \sqsubseteq \{a, c\}$ follows from the other AD formulas. Following Algorithm 2, we take the complement of $\{a, c\}$ and compute the greatest model of the remaining set of the formulas that is smaller than or equal to the complement of $\{a, c\}$. Then, we conclude that $\{f, g\} \sqsubseteq \{a, c\}$ is entailed by the rest of the formulas, and therefore, redundant, if and only if $\{f, g\}$ is disjoint with the computed model.

The complement of $\{a, c\}$ is $\{b, d, e, f, g\}$. Denote the complement by N. We now compute the greatest model according to Algorithm 1. Taking the first AD formula $\{a, b\} \sqsubseteq \{c\}$, we see $b \in N \cap \{a, b\}$, but $c \notin N$, i.e., b is to be removed from N. The second formula $\{c, d\} \sqsubseteq \{a\}$ makes d removed from $N = \{d, e, f, g\}$ because $d \in N$ and $a \notin N$. The third formula $\{c, e\} \sqsubseteq \{a, d\}$ makes e removed from $N = \{e, f, g\}$ because $\{a, d\} \cap N = \emptyset$. The fourth formula is omitted because it is the formula for which we test redundancy. The fifth formula $\{d, f\} \sqsubseteq \{b, c\}$ makes f removed from $N = \{f, g\}$ because $f \in N$ and $\{b, c\} \cap N = \emptyset$. Finally, the last formula $\{g\} \sqsubseteq \{e, f\}$ makes g removed from $N = \{g\}$ because $g \in N$ and $\{e, f\} \cap N = \emptyset$. Therefore, we have arrived at $N = \emptyset$, i.e., $N \cap \{f, g\} = \emptyset$. According to Theorem 7, this means that $\{f, g\} \sqsubseteq \{a, c\}$ follows from the other AD formulas.

IV. Conclusion and Future Research

The main goal of this paper is to emphasize the need for taking into account background knowledge in FCA. A background knowledge represents an additional information regarding the input data that a user may have. Such information can be used in the process of FCA to define what is interesting for the user. Particularly, a background knowledge can serve as a constraint specifying which formal concepts are interesting, and should therefore be extracted from data and presented to the user, and which are not. This may significantly reduce the number of formal concepts extracted from the input data. Using background knowledge thus enables a focused extraction of knowledge and may considerably reduce the amount of information presented to the user. Such a reduction is the main practical effect of using background knowledge in FCA.

In particular, we presented an approach to representation and treatment of background knowledge that concerns user's priorities regarding attributes and their relative importance. We focused on the main notions and conceptual issues involved, and provided theoretical foundations, algorithms, and illustrative examples.

Future research will focus on further methodological and theoretical development of using background knowledge in FCA as well as on applications of FCA with background knowledge. The main topics for future research are the following.

1) *Interactive specification of background knowledge*: AD formulas representing constraints can be constructed after

the user answers a series of questions like "do you prefer A over B?" By showing the user just the concepts congruent with the priorities specified so far, we draw his attention to the groups of objects, e.g., products that are coherent with user's expectations. During the exploration of the congruent formal concepts, the user can eventually come up with new relevant constrains that were not explicitly mentioned in the beginning but were subconsciously expected by the user. This way, the exploration based on constrained concept lattices may help the user further clarify his feeling of properties that are important for the particular task of interest. Investigation of such interactive procedures for data exploration based on user priorities is a topic for future research.

2) *Development of efficient algorithms for computing the set of formal concepts compatible with background knowledge*: Efficient algorithms are expected to compute directly the formal concepts that are compatible with the background knowledge, without necessarily computing all formal concepts.
3) *Theoretical development*: We have demonstrated that theoretical insight is crucial for design of efficient algorithms. Further theoretical analysis is necessary. As an example, an analogy of the basic theorem for concept lattices in case when background knowledge is taken into account is an open problem.

REFERENCES

[1] R. Belohlavek and V. Sklenar, "Formal concept analysis constrained by attribute-dependency formulas," in *Proc. ICFCA 2005* (Lecture Notes in Computer Science), vol. 3403. New York: Springer-Verlag, pp. 176–191.

[2] R. Belohlavek and V. Vychodil, "Formal concept analysis with constraints by closure operators," in *Proc. ICCS 2006* (Lecture Notes in Artificial Intelligence), vol. 4068. New York: Springer-Verlag, pp. 131–143.

[3] R. Belohlavek and V. Vychodil, "Reducing the size of fuzzy concept lattices by fuzzy closure operators," in *Proc. SCIS, ISIS 2006*, Tokyo, Japan, pp. 309–314.

[4] R. Belohlavek and V. Vychodil, "Semantic entailment of attribute-dependency formulas and their non-redundant bases," in *Proc. JCIS 2006*, pp. 747–750.

[5] R. Belohlavek and V. Vychodil, "Adding background knowledge to formal concept analysis via attribute dependency formulas," in *Proc. 23rd Annu. ACM Symp. Appl. Comput. 2008*, pp. 938–943.

[6] J.-F. Boulicaut and B. Jeudy, "Constraint-based data mining," in *The Data Mining and Knowledge Discovery Handbook*. New York: Springer-Verlag, 2005, pp. 399–416.

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

[8] I. Davidson and S. S. Ravi, "Hierarchical clustering with constraints: Theory and practice," in *Proc. PKDD 2005* (Lecture Notes in Artificial Intelligence), vol. 3721. New York: Springer-Verlag, pp. 59–70.

[9] J.-P. Doignon and J.-C. Falmagne, *Knowledge Spaces*. Berlin, Germany: Springer-Verlag, 1999.

[10] B. S. Everitt, *Cluster Analysis*, 4th ed. London, U.K.: E. Arnold, 2001.

[11] B. Ganter, "Attribute exploration with background knowledge," *Theor. Comput. Sci.*, vol. 217, pp. 215–233, 1999.

[12] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*. Berlin, Germany: Springer-Verlag, 1999.

[13] J.-L. Guigues and V. Duquenne, "Familles minimales d'implications informatives resultant d'un tableau de données binaires," *Math. Sci. Hum.*, vol. 95, pp. 5–18, 1986.

[14] J. W. Lloyd, *Foundations of Logic Programming*, 2nd ed. New York: Springer-Verlag, 1987.

[15] D. Maier, *The Theory of Relational Databases*. Rockville, MD: Comput. Sci., 1983.

[16] G. Stumme, "Attribute exploration with background implications and exceptions," in *Data Analysis and Information Systems: Statistical and Conceptual Approaches*. New York: Springer-Verlag, 1996, pp. 457–469.

[17] K. Wagsta, C. Cardie, S. Rogers, and S. Schroedl, "Constrained K-means clustering with background knowledge," in *Proc. ICML 2001*, Williamstown, MA, pp. 577–584.

Radim Belohlavek (SM'07) received the Graduate degree in theoretical cybernetics and informatics and systems science from Palacky University, Olomouc, Czech Republic, the M.Sc. degree (*summa cum laude*) in 1994, two Ph.D. degrees in computer science and mathematics from Technische Universität Ostrava, Ostrava, Czech Republic and Palacky University, Olomouc, Czech Republic, in 1998 and 2001, respectively, and the D.Sc. degree in computer science and cybernetics from the Academy of Sciences of the Czech Republic, Prague, Czech Republic, in 2008.

From 2001 to 2007, he was the Head of the Department of Computer Science, Palacky University. In 2005, he was appointed as a Full Professor of Computer Science by the President of the Czech Republic. Since 2007, he has been a Professor of systems science at the State University of New York at Binghamton, Binghamton. His current research interests include areas of uncertainty and information, logic and algebra, fuzzy logic and fuzzy sets, and relational data analysis. He was the principal investigator of numerous grants in these areas. He has authored or coauthored two monographs (Kluwer, Springer) and over 130 papers published in conference proceedings and journals.

Prof. Belohlavek is a member of the Association for Computing Machinery (ACM) and the American Mathematical Society (AMS).

Vilem Vychodil (M'06) received the Graduate degree in computer science the M.Sc. degree (*summa cum laude*) in 2002, and the Ph.D. degree in mathematics in 2004, all from Palacky University, Olomouc, Czech Republic.

From 2000 to 2007, he was with the Department of Computer Science, Palacky University. Since 2007, he has been with the Department of Systems Science and Industrial Engineering, State University of New York at Binghamton, Binghamton. His current research interests include fuzzy logic, fuzzy relational systems, relational data analysis, uncertainty in data, mathematical logic, and logical foundations of knowledge engineering. He has authored or coauthored one monograph (Springer) and over 70 papers published in conference proceedings and journals.

Dr. Vychodil is a member of the Association for Computing Machinery (ACM).