Closure-based constraints in formal concept analysis

Radim Belohlavek, Vilem Vychodil

Data Analysis and Modeling Laboratory (DAMOL),
Dept. Computer Science, Palacky University, Olomouc, Czech Republic;
e-mail: radim.belohlavek@acm.org, vychodil@acm.org.

Abstract

We present a method of imposing constraints in extracting formal concepts (equivalently, closed itemsets or fixpoints of Galois connections) from a binary relation. The constraints are represented by closure operators and their purpose is to mimic background knowledge a user may have of the data. The idea is to consider relevant and thus to extract only these itemsets that are compatible with the background knowledge. As a result, the method extracts less clusters, those that are interesting from the user point of view, in a shorter time. The method makes it also possible to extract minimal bases of attribute dependencies in the input data that are compatible with the background knowledge. We provide examples of several particular types of constraits including those that appeared in the literature in the past and present algorithms to compute the constrained formal concepts and attribute implications.

Keywords: closure structure, binary relation, formal concept analysis, attribute implication, background knowledge, algorithms

1. Introduction and Motivation

Formal concept analysis (FCA [7, 9]) is a method of data analysis and visualization which deals with object-attribute input data and aims at extracting from the data particular conceptual clusters (so-called formal concepts) and data dependencies (so-called attribute implications). The input data is in the form of a table describing objects (rows), their attributes (columns), and their "yes/no" relationship. The presence/absence of attributes is indicated by $\times$'s or blanks in the table, see Fig. 1. The set of all formal concepts in the input data forms a complete lattice. This lattice, called a concept lattice, is utilized for the purpose of data analysis [9]. Another output utilized for data analysis is a minimal basis [9] of all attribute implications that are valid in the input data. FCA proved to be useful in several areas, both as a direct method of data analysis, see e.g. [7, 9] and the references therein, and as a method for data preprocessing, see e.g. [22]. Recently, it has been shown in [6] that formal concepts can be used to find optimal factorization of Boolean matrices and can be used as optimal solutions to the discrete basis problem discussed in [17]. Note also

that in terms of association rule mining, (intents of) formal concepts coincide with closed itemsets. Extracting formal concepts is therefore an important task.

One of the main challenges in FCA results from the fact that the number of formal concepts of even a middle-size data is usually large. If a concept lattice is to be directly examined by a user, its large size poses a problem. There exist various approaches to overcome this problem, namely those based on factorizations and decompositions of large concept lattices [9]. In this paper, we present a different approach. The approach is based on constraining concept lattices by an additional information which a user may have about the input data. In the basic setting of FCA, it is assumed that no further information is supplied at the input except for the data table. However, it is often the case that there is an additional information about the data available—a user background knowledge. A background knowledge may naturally serve as a constraint. Namely, the background knowledge can be used to filter out formal concepts which do not comply with it (these are ruled out as "noninteresting"), while keeping (i.e., presenting to the user) formal concepts compatible with it (the "interesting" ones). This way, the number of extracted formal concepts is reduced by focusing on the "interesting" ones. The same idea applies if one is interested in discovering attribute dependencies in the data—instead of discovering a large (and less useful) set of dependencies from the whole data, one can focus on discovering dependencies that are compatible with the background knowledge. One can then expect that such targeted dependencies are more informative for the user who supplied the background knowledge.

In this paper, we develop an approach of imposing constraints in FCA which are represented by means of closure operators. The approach covers several natural kinds of constraints. For instance, one can set the closure operator in such a way that the conceptual clusters satisfying the constraint correspond exactly to closed frequent itemsets [19] used, e.g., in generating non-redundant association rules [20, 22], see also Section 7.

The paper is organized as follows. In Section 2, we present preliminaries from FCA. In Section 3, we deal with constraining concept lattices. In Section 4, an algorithm for computing constrained concept lattices is described. In Section 5, we present a method to constrain attribute implications, describe non-redundant bases of constrained attribute implications, and present an algorithm that computes the bases. In Section 6, we provide a syntactico-semantically complete axiomatization for the constrained attribute implications. Section 7 contains several examples of constraints by closure operators and illustrative examples. In Section 8, we deal with combinations of constrains which allow us to create compound constraints from elementary ones. Section 9 contains a summary.

2. Preliminaries

In this section, we summarize the basic notions of FCA. For a detailed information, we refer to [7, 9].

An object-attribute data table describing which objects have which attributes (features) can be identified with a triplet $\langle X, Y, I \rangle$ where X is a finite non-empty

set (of objects), Y is a finite non-empty 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 $\times$; if $\langle x, y\rangle \notin I$ the table entry contains blank symbol). In terms of FCA, the triplet $\langle X, Y, I\rangle$ is called a formal context.

Example 1. Fig. 1 represents a formal context with food products as objects and food additives as attributes. The object attribute relation $I \subseteq X \times Y$ indicates whether a food product $x \in X$ does or does not contain an additive $y \in Y$. The objects are denoted by numbers, i.e., $X = \{1, \ldots, 27\}$, and the attributes are denoted by letters, i.e., $Y = \{a, \ldots, f\}$. We use this data for illustrative purposes in this paper. ■

Every data table $\langle X, Y, I\rangle$ induces a pair of so-called concept-forming operators. For each $A \subseteq X$ and $B \subseteq Y$ denote by $A^{\uparrow_I}$ a subset of Y and by $B^{\downarrow_I}$ a subset of X defined by

$$A^{\uparrow_I} = \{y \in Y \mid \text{for each } x \in A\colon \langle x, y\rangle \in I\}, \tag{1}$$

$$B^{\downarrow_I} = \{x \in X \mid \text{for each } y \in B\colon \langle x, y\rangle \in I\}. \tag{2}$$

That is, $A^{\uparrow_I}$ is the set of all attributes from Y shared by all objects from A and $B^{\downarrow_I}$ is the set of all objects from X having all attributes from B. A formal concept in $\langle X, Y, I\rangle$ is a pair $\langle A, B\rangle$, consisting of $A \subseteq X$ and $B \subseteq Y$, that satisfies $A^{\uparrow_I} = B$ and $B^{\downarrow_I} = A$. Formal concepts are thought of as conceptual clusters in the input data. If I is obvious, we abbreviate $^{\uparrow_I}$ and $^{\downarrow_I}$ by $^{\uparrow}$ and $^{\downarrow}$, respectively.

Remark 1. By definition, a formal concept $\langle A, B\rangle$ consists of a set A (so-called extent) of objects which fall under the concept and a set B (so-called intent) of attributes which 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 $\langle X, Y, I\rangle$ which are full of $\times$'s: for $A \subseteq X$ and $B \subseteq Y$, $\langle A, B\rangle$ is a formal concept in $\langle X, Y, I\rangle$ iff $A \times B \subseteq I$ and there is no $A' \supset A$ or $B' \supset B$ such that $A' \times B \subseteq I$ or $A \times B' \subseteq I$. See [9] for more details. In a data like that in Fig. 1, there may exist formal concepts representing "dairy products", "frozen food products", "soups", "breakfast cereals", and the like. ■

The set of all concepts in $\langle X, Y, I\rangle$ is denoted by

$$\mathcal{B}(X, Y, I) = \{\langle A, B\rangle \mid A^{\uparrow} = B \text{ and } B^{\downarrow} = A\}. \tag{3}$$

$\mathcal{B}(X, Y, I)$ can be equipped with a partial order $\leq$, modeling a subconcept-superconcept hierarchy, defined by

$$\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle \text{ iff } A_1 \subseteq A_2 \text{ (iff } B_2 \subseteq B_1). \tag{4}$$

		E322 a	E330 b	E440 c	E471 d	E476 e	E500 f
dia muesli	1	×	×		×		×
cherry muesli	2	×	×		×	×	×
chocolate muesli	3				×		×
strawberry yogurt	4	×	×	×			
hazelnut wafers	5	×			×	×	×
lemon soda	6		×	×			
stracciatella yogurt	7	×		×			
chocolate soy bar	8	×		×		×	
milky way bar	9	×	×	×			×
assorted chocolates	10	×	×			×	
choc-ice	11	×			×		
cranberry muesli	12	×	×				×
margarine 1	13		×		×	×	
enriched margarine	14		×		×		
gingerbread	15	×	×	×		×	×
margarine 2	16	×	×				
margarine 3	17	×	×		×	×	
margarine 4	18	×	×		×		
hazelnut chocolate	19	×				×	
raspberry jelly	20			×			
raisin chocolate	21	×	×	×		×	
cinnamon cookies	22						×
wafers	23	×					×
vegetable broth	24				×		
chocolate wafers	25	×				×	×
chicken broth	26		×				
dia ginger cookies	27	×					

Figure 1: Object-attribute data table. Legend: objects = food products, attributes = food additives (E322: lecithin; E330: citric acid; E440: pectin; E471: mono- and diglycerides; E476: polyglycerol polyricinoleate; E500: sodium carbonates).

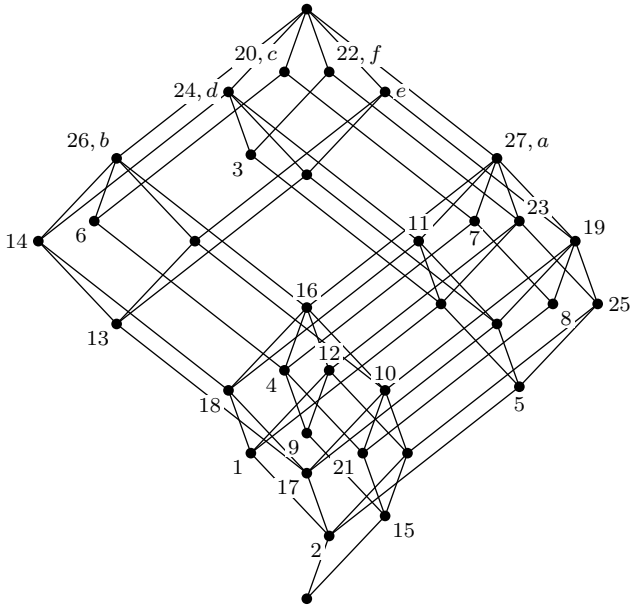

Figure 2: Concept lattice for data from Fig. 1.

Thus, $\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle$ means that concept $\langle A_1, B_1\rangle$ is less general (i.e., more specific) than concept $\langle A_2, B_2\rangle$ in that it covers less objects and more attributes. Note that $^{\uparrow}$ and $^{\downarrow}$ defined by (1) and (2) form a Galois connection [9] and that $\mathcal{B}(X, Y, I)$ is in fact the set of all fixed points of $^{\uparrow}$ and $^{\downarrow}$. Considering a food products data, the hierarchy of its concepts can comprise information that a concept of "pastry" is more general (i.e., less specific) than a concept of "breads" and that the concept of "breads" is more general (less specific) than a concept of "sourdough breads".

$\mathcal{B}(X, Y, I)$ equipped with $\leq$ happens to be a complete lattice (i.e., a partially ordered set where every set of concepts has its infimum and supremum), called a concept lattice of $\langle X, Y, I\rangle$, the basic structure of which is described by the so-called Basic Theorem of FCA [9]:

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

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

(*ii*) *An arbitrary complete lattice $\langle V, \leq\rangle$ is isomorphic to $\mathcal{B}(X, Y, I)$ iff there are maps $\gamma\colon X \to V$, $\mu\colon Y \to V$ where*

- *$\gamma(X)$ is $\bigvee$-dense in V, $\mu(Y)$ is $\bigwedge$-dense in V, and*
- *$\gamma(x) \leq \mu(y)$ iff $\langle x, y\rangle \in I$.* □

Note that part (*ii*) of Theorem 1 provides a way to visualize and label concept lattices [9]: one depicts the clusters as nodes (vertices) and the subconcept-superconcept relationship as edges in the line diagram of the lattice. The labeling allows us to identify extents and intents of all concepts. Namely, 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 according to (4) which contains x (i.e., if $A = \{x\}^{\uparrow\downarrow}$ and $B = \{x\}^{\uparrow}$); 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 according to (4) which contains y (i.e., if $A = \{y\}^{\downarrow}$ and $B = \{y\}^{\downarrow\uparrow}$). As a consequence, both the extent A and the intent B of a 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.

Example 2. Consider the data $\langle X, Y, I\rangle$ from Fig. 1. The corresponding concept lattice $\mathcal{B}(X, Y, I)$ is depicted in Fig. 2. The concept lattice contains 35 concepts denoted by black nodes in the diagram. Edges correspond to the subconcept-superconcept ordering (4). The concept lattice contains two trivial concepts. Namely, the bottom node corresponds to a concept $\langle \emptyset, Y\rangle$ (concept with no objects); the top-most node corresponds to a concept $\langle X, \emptyset\rangle$ (concept of all objects having no common attributes). Except for these two borderline concepts, we have 33 nontrivial ones. Consider now the node labeled by 7. The node represents a concept $\langle A, B\rangle = \langle\{4, 7, 8, 9, 15, 21\}, \{a, c\}\rangle$. Indeed, if we go from the node "upwards" along the edges, we can see that the node contains attributes a (lecithin) and c (pectin). If we go "downwards", we see that the concept consists of objects 4 (strawberry yogurt), 8 (chocolate soy bar), 9 (milky way bar), 21 (raisin chocolate), and 15 (gingerbread). Since the node is labeled by 7, it also contains object 7 (stracciatella yogurt). As one can see, the concept can be interpreted as a cluster of exactly all products containing lecithin and pectin. ■

The other patterns extraced in FCA from the input data are attribute implications [9, 10]. An attribute implication (over a set Y of attributes) is an expression $A \Rightarrow B$ where $A, B \in 2^Y$ are sets of attributes. $A \Rightarrow B$ is valid in the input data $\langle X, Y, I\rangle$ if whenever an object has all the attributes from A, then it has all the attributes from B as well. The basic notions concerning attribute implications are summarized below. We say that $A \Rightarrow B$ is valid in a set $M \subseteq Y$ of attributes, written $M \models A \Rightarrow B$, if the following condition is satisfied:

$$\text{if } A \subseteq M \text{ then } B \subseteq M. \tag{5}$$

Hence, if M represents the set of attributes of an object x then $M \models A \Rightarrow B$ means that if x has all the attributes from A then it has all the attributes from B. More generally, for any $\mathcal{M} \subseteq 2^Y$, we put $\mathcal{M} \models A \Rightarrow B$ if $M \models A \Rightarrow B$ for each $M \in \mathcal{M}$. Therefore, the notion of validity in input data may be rephrased as follows: $A \Rightarrow B$ is valid in $\langle X, Y, I\rangle$ if and only if $\mathcal{M} \models A \Rightarrow B$ for $\mathcal{M} = \{\{x\}^{\uparrow} \mid x \in X\}$ (notice that $\{x\}^{\uparrow}$ is the set of all attributes of object x). The fact that $A \Rightarrow B$ is valid in $\langle X, Y, I\rangle$ is denoted by $I \models A \Rightarrow B$.

Attribute implications valid in formal contexts may conveniently be represented by certain small sets called bases. Given a set T of attribute implications, $M \subseteq Y$ is called a model of T if $M \models A \Rightarrow B$ for each $A \Rightarrow B$ from T. The system of all models of T is denoted by $\mathrm{Mod}(T)$. An attribute implication $A \Rightarrow B$ is (semantically) entailed by T, written $T \models A \Rightarrow B$, if $\mathrm{Mod}(T) \models A \Rightarrow B$ (i.e., $A \Rightarrow B$ is valid in every model of T). A set T of attribute implications is called complete in $\langle X, Y, I \rangle$ if for every implication $A \Rightarrow B$ we have: $I \models A \Rightarrow B$ ($A \Rightarrow B$ is true in input data) if and only if $T \models A \Rightarrow B$ ($A \Rightarrow B$ is entailed by T). Furthermore, T is called a base of $\langle X, Y, I \rangle$ if T is complete in $\langle X, Y, I \rangle$ and no proper subset of T is complete in $\langle X, Y, I \rangle$. Thus, a base of $\langle X, Y, I \rangle$ is a non-redundant set of attribute implications valid in $\langle X, Y, I \rangle$ that entails exactly all implications valid in $\langle X, Y, I \rangle$. In this sense, a base is a minimal fully informative set of attribute implications valid in the input data. More details including algorithms for finding bases can be found in [9, 10].

Example 3. The data $\langle X, Y, I \rangle$ from Fig. 1 has a minimal base containing five attribute implications:

$$\{e, f\} \Rightarrow \{a, e, f\}, \qquad \{c, f\} \Rightarrow \{a, b, c, f\}, \qquad \{c, e\} \Rightarrow \{a, c, e\},$$
$$\{c, d\} \Rightarrow Y, \qquad \{b, f\} \Rightarrow \{a, b, f\}.$$

Therefore, all attribute implications valid in $\langle X, Y, I \rangle$ are exactly the attribute implications entailed by the base of five implications. This paritcular base is a so-called Guigues-Duquenne base [10]. The base can be further simplified by removing attributes from consequents of the implications. For instance,

$$\{e, f\} \Rightarrow \{a\}, \qquad \{c, f\} \Rightarrow \{a, b\}, \qquad \{c, e\} \Rightarrow \{a\},$$
$$\{c, d\} \Rightarrow \{a, b, e, f\}, \qquad \{b, f\} \Rightarrow \{a\},$$

is another base of $\langle X, Y, I \rangle$. Put in words, the first implication $\{e, f\} \Rightarrow \{a\}$ says that "if a product contains polyglycerol polyricinoleate and sodium carbonates then it contains lecithin as well". The other implications can be interpreted in a similar way.

Note that Guigues-Duquenne bases are minimal in terms of their size. In the above example, it means that $\langle X, Y, I \rangle$ does not have a base consisting of four or less attribute implications [10].

3. Formalization of Constraints

Selecting "interesting" concepts from the set $\mathcal{B}(X, Y, I)$ of all formal concepts needs to be based on a criterion of interestingness. Such a criterion can be seen as a constraint represented by a user background knowledge and needs to be supplied by a user along with the input data $\langle X, Y, I \rangle$.

One way to specify "interesting" concepts, which is explored in this paper, is to define them as concepts with "interesting intents" (i.e., interesting sets of attributes covered by the concepts). Thus, given a data table $\langle X, Y, I \rangle$, the user

may specify a system $\mathcal{S} \subseteq 2^Y$ of subsets of Y. A concept $\langle A, B\rangle \in \mathcal{B}(X, Y, I)$ is then considered "interesting" if $B \in \mathcal{S}$. Several important issues have to be resolved, however. First, which systems $\mathcal{S} \subseteq 2^Y$ lead to constrained concepts which form a hierarchically-ordered substructure of $\mathcal{B}(X, Y, I)$ that (somehow) approximates the original hierarchy? Second, how should $\mathcal{S}$ be described by a user? Third, how to determine the constrained concept lattice without the need to compute the original (large) concept lattice first? Fourth, can we determine and compute a minimal set of rules describing if-then dependencies among attributes of the constrained concept lattice? In this paper, we develop the above-mentioned approach to constraints and answer the foregoing questions provided that the sets of attributes which are considered "interesting" form a closure system in Y. As we see in Section 7, this choice covers a large family of particular constraints and includes some approaches presented in the literature in the past. (A reader who is only interested in examples may continue reading this section up to Definition 2 and then skip to Section 7.)

Recall that $\mathcal{S} \subseteq 2^Y$ is called a closure system in Y if $\mathcal{S}$ is closed under arbitrary intersections. A closure operator in a set Y is a map $C\colon 2^Y \to 2^Y$ satisfying

$B \subseteq C(B)$,	extensivity
$B_1 \subseteq B_2$ implies $C(B_1) \subseteq C(B_2)$,	monotony
$C(C(B)) = C(B)$,	idempotency

for any $B, B_1, B_2 \in 2^Y$. A set $B \subseteq Y$ such that $C(B) = B$ is called a fixed point of C. The set of all fixed points of C is denoted by $\mathrm{fix}(C)$. For each closure system $\mathcal{S} \subseteq 2^Y$ one can define an operator $C_{\mathcal{S}}$ by $C_{\mathcal{S}}(B) = \bigcap\{A \in \mathcal{S} \mid B \subseteq A\}$. It is a well-known fact that (i) $\mathrm{fix}(C)$ is a closure system in Y, (ii) $C_{\mathcal{S}}$ is a closure operator in Y, and (iii) there is a one-to-one correspondence between closure operators in Y and closure systems in Y: we have $C = C_{\mathrm{fix}(C)}$ and $\mathcal{S} = \mathrm{fix}(C_{\mathcal{S}})$, i.e. the maps $C \mapsto \mathrm{fix}(C)$ and $\mathcal{S} \mapsto C_{\mathcal{S}}$ are mutually inverse, see [9] for details.

We now formalize "interesting sets of attributes" as follows:

Definition 1. Let Y be a set of attributes, C be a closure operator in Y. A set $B \subseteq Y$ is called a *C-interesting set of attributes* (shortly, a *set of C-attributes*) if $B = C(B)$. ■

Throughout the paper, Y denotes a finite set of attributes and $C\colon 2^Y \to 2^Y$ denotes a closure operator in Y. Put verbally, Definition 1 says that C-interesting sets of attributes are exactly the fixed points of the closure operator C. Thus, given any set $B \subseteq Y$ of attributes, $C(B)$ can be seen as *the least set of C-interesting attributes containing B.*

Remark 2. (i) Given a set $B \subseteq Y$ of attributes, either we have $B = C(B)$, i.e. B is C-interesting, or $B \subset C(B)$ which can be read: "B is not C-interesting, but additional attributes $C(B) - B$ would make B interesting". Thus, C describes which set of attributes are interesting and which attributes must be added to a set of attributes to make it interesting.

(*ii*) A definition of C depends on a particular application. In our approach, we assume that C may be an arbitrary closure operator, covering thus all possible choices of C. Obviously, in real applications, it is necessary to have a collection of easy-to-understand definitions of such closure operators. In Section 7 and Section 8 we provide several examples of such operators.

(*iii*) Clearly, constraints represented by closure operators do not exhaust all possible types of user constraints. For example, a different type of constraints, representing relative importance of attributes, is investigated in [4]. Nevertheless, as we show in Section 7, constraints represented by closure operators include a variety of particular natural constraints. ■

Definition 2. Let $\langle X, Y, I \rangle$ be a data table, C be a closure operator in Y. We put

$$\mathcal{B}_C(X, Y, I) = \{\langle A, B \rangle \in \mathcal{B}(X, Y, I) \,|\, B = C(B)\}. \tag{6}$$

Each $\langle A, B \rangle \in \mathcal{B}_C(X, Y, I)$ is called a *C-interesting concept* (shortly, a *C-concept*) in $\langle X, Y, I \rangle$. If $\langle A, B \rangle \in \mathcal{B}_C(X, Y, I)$ is a C-concept then A and B are called a *C-interesting extent* (*C-extent*) and a *C-interesting intent* (*C-intent*), respectively. The sets of all C-extents and C-intents are denoted by $\mathrm{Ext}_C(X, Y, I)$ and $\mathrm{Int}_C(X, Y, I)$, respectively. ■

Remark 3. (*i*) According to Definition 2, $\langle A, B \rangle$ is a C-concept iff $\langle A, B \rangle$ is a concept (in the ordinary sense) such that B is a set of C-attributes. Hence, $\langle A, B \rangle$ is a C-concept iff $A^{\uparrow} = B$, $B^{\downarrow} = A$, and $C(B) = B$ hold. As a consequence, C-concepts $\langle A, B \rangle$ can be seen as maximal rectangles in the input data table which are full of $\times$'s, see Section 2, such that B is closed under C.

(*ii*) Notice that two boundary cases of closure operators in Y are $C_0(B) = B$ (for all $B \subseteq Y$) and $C_1(B) = Y$ (for all $B \subseteq Y$). The notion of a C_0-concept coincides with that of a concept. Hence, $\mathcal{B}_{C_0}(X, Y, I)$ equals $\mathcal{B}(X, Y, I)$. In case of C_1, $\mathcal{B}_{C_1}(X, Y, I)$ is a one-element set containing $\langle Y^{\downarrow}, Y \rangle$.

(*iii*) Observe that $B \subseteq Y$ is a C-intent iff $B = B^{\downarrow\uparrow} = C(B)$. Thus, $\mathrm{Int}_C(X, Y, I) = \mathrm{Int}(X, Y, I) \cap \mathrm{fix}(C)$, where $\mathrm{Int}(X, Y, I)$ denotes the set of all intents. ■

The following assertion characterizes the structure of C-concepts and shows that $\mathcal{B}_C(X, Y, I)$ equipped with the subconcept-superconcept hierarchy is a particular substructure of $\mathcal{B}(X, Y, I)$. Note that a complete $\bigvee$-sublattice of $\mathcal{B}(X, Y, I)$ is a subset of $\mathcal{B}(X, Y, I)$ which is closed under arbitrary suprema.

Theorem 2. *Let $\langle X, Y, I \rangle$ be a data table, C be a closure operator. Then $\mathcal{B}_C(X, Y, I)$ equipped with $\leq$ defined by* (4) *is a complete $\bigvee$-sublattice of $\mathcal{B}(X, Y, I)$.*

Proof. Take an arbitrary J-indexed system

$$S = \{\langle A_j, B_j \rangle \in \mathcal{B}_C(X, Y, I) \,|\, j \in J\}$$

of C-concepts. We prove that

$$\langle A, B \rangle = \left\langle \left(\bigcup_{j \in J} A_j\right)^{\uparrow\downarrow}, \bigcap_{j \in J} B_j \right\rangle \tag{7}$$

is the least upper bound of S in $\mathcal{B}_C(X,Y,I)$ which will prove that $\mathcal{B}_C(X,Y,I)$ is a complete $\bigvee$-sublattice of $\mathcal{B}(X,Y,I)$, see Theorem 1. First, since $B_j \in \mathrm{fix}(C)$ is true for each $j \in J$, it follows that $\bigcap_{j\in J} B_j \in \mathrm{fix}(C)$. Therefore, $\langle A, B\rangle$ defined by (7) belongs to $\mathcal{B}_C(X,Y,I)$. Moreover, since $B \subseteq B_j$ for arbitrary $j \in J$, we get $\langle A_j, B_j\rangle \leq \langle A, B\rangle$ according to (4). Thus, $\langle A, B\rangle$ is an upper bound of S in $\mathcal{B}_C(X,Y,I)$. In addition to that, if there is $\langle A', B'\rangle \in \mathcal{B}_C(X,Y,I)$ such that $\langle A_j, B_j\rangle \leq \langle A', B'\rangle$ for all $j \in J$, then $B' \subseteq B_j$ for all $j \in J$ and thus, $B' \subseteq \bigcap_{j\in J} B_j = B$, meaning that $\langle A, B\rangle \leq \langle A', B'\rangle$, i.e., $\langle A, B\rangle$ is indeed the least upper bound (a supremum) of S in $\mathcal{B}_C(X,Y,I)$. According to Theorem 1, this supremum agrees with the supremum of S in $\mathcal{B}(X,Y,I)$. □

The following theorem shows that conversely, every $\bigvee$-sublattice of $\mathcal{B}(X,Y,I)$ can be seen as a set of all C-concepts for a suitable operator C.

Theorem 3. *Let $\mathcal{B}(X,Y,I)$ be a concept lattice and let $\mathcal{L}$ be a complete $\bigvee$-sublattice of $\mathcal{B}(X,Y,I)$. Then, $\mathcal{L}$ equals $\mathcal{B}_C(X,Y,I)$ for some closure operator C.*

Proof. Put $\mathcal{S} = \{B \subseteq Y \mid \langle A, B\rangle \in \mathcal{L}\}$. Observe that $\mathcal{S}$ is a closure system in Y. Indeed, take arbitrary $\langle A_j, B_j\rangle \in \mathcal{L}$ $(j \in J)$. Since $\mathcal{L}$ is a complete $\bigvee$-sublattice of $\mathcal{B}(X,Y,I)$, the supremum of all $\langle A_j, B_j\rangle$ $(j \in J)$ in $\mathcal{L}$ coincides with the supremum of all $\langle A_j, B_j\rangle$ $(j \in J)$ in $\mathcal{B}(X,Y,I)$. Applying Theorem 1, the latter yields $\bigcap_{j\in J} B_j \in \mathcal{S}$. Consider the closure operator $C_{\mathcal{S}}$ induced by $\mathcal{S}$. Since $\mathrm{fix}(C_{\mathcal{S}}) = \mathcal{S}$, one can conclude that $\mathcal{L}$ equals $\mathcal{B}_C(X,Y,I)$ with C being $C_{\mathcal{S}}$. □

Putting previous two assertions together, we get

Corollary 1. *Let $\mathcal{B}(X,Y,I)$ be a concept lattice and let $\mathcal{L} \subseteq \mathcal{B}(X,Y,I)$. Then the following are equivalent:*

- *(i) $\mathcal{L}$ is a set of C-concepts for some C.*
- *(ii) $\mathcal{L}$ equipped with partial order $\leq$ defined by (4) is a complete $\bigvee$-sublattice of $\mathcal{B}(X,Y,I)$.*

Remark 4. Notice that each $\mathcal{B}_C(X,Y,I)$ contains $\langle Y^{\downarrow}, Y\rangle$, i.e. the least concept of $\mathcal{B}(X,Y,I)$. This follows from the extensivity of C. Therefore, $\langle Y^{\downarrow}, Y\rangle$ is the least C-concept of $\mathcal{B}_C(X,Y,I)$, see (4). This might seem to be in contradiction with our intuition about "interesting concepts" because $\langle Y^{\downarrow}, Y\rangle$ has all attributes and is thus trivial. The presence of $\langle Y^{\downarrow}, Y\rangle$ is to be understood as a technical condition that guarantees that $\mathcal{B}_C(X,Y,I)$ is a complete lattice. ■

4. Algorithms for Computing $\mathcal{B}_C(X,Y,I)$

We now turn our attention to the computational aspects of enumerating all C-concepts., i.e. all elements of $\mathcal{B}_C(X,Y,I)$. A naive way to compute $\mathcal{B}_C(X,Y,I)$ is to compute $\mathcal{B}(X,Y,I)$ first and then go through all of its concepts and list all the C-concepts. This method may not be efficient because $\mathcal{B}(X,Y,I)$

may be very large compared to the size of $\mathcal{B}_C(X, Y, I)$. In this section, we show that $\mathcal{B}_C(X, Y, I)$ can be directly computed without the need to enumerate all concepts from $\mathcal{B}(X, Y, I)$.

We are going to reduce the problem of computing C-concepts to the problem of computing fixed points of a single closure operator. Recall that the composition ${}^{\downarrow\uparrow} : 2^Y \to 2^Y$ of the concept-forming operators (1) and (2) is a closure operator [9]. Since C is assumed to be a closure operator as well, C-interesting concepts are exactly the common fixed points of ${}^{\downarrow\uparrow}$ (operator induced by the Galois connection given by a formal context $\langle X, Y, I\rangle$) and C (operator specifying interesting sets of attributes). Thus, we may define a composite operator as follows.

For any $B \subseteq Y$ define sets $B_j \subseteq Y$ ($j \in \mathbb{N}_0$) by

$$B_j = \begin{cases} B, & \text{if } j = 0, \\ C({B_{j-1}}^{\downarrow\uparrow}), & \text{if } j \geq 1. \end{cases} \tag{8}$$

Using (8), define an operator $cl\colon 2^Y \to 2^Y$ by

$$cl(B) = \textstyle\bigcup_{j=0}^{\infty} B_j. \tag{9}$$

Theorem 4. *Let $\langle X, Y, I\rangle$ be a formal context, C be a closure operator in Y. Then $cl : 2^Y \to 2^Y$ defined by (9) is a closure operator such that $\mathrm{fix}(cl) = \mathrm{Int}_C(X, Y, I)$, i.e. the fixed points of cl are just the C-intents.*

Proof. Since both ${}^{\downarrow\uparrow}$ and C are closure operators, $B_0 \subseteq B_1 \subseteq \cdots$, and $B_j \subseteq cl(B)$ for each $j \in \mathbb{N}_0$. Furthermore, the extensivity and monotony of ${}^{\downarrow\uparrow}$ and C yield extensivity and monotony of cl. To check idempotency of cl, we show $C((cl(B))^{\downarrow\uparrow}) \subseteq cl(B)$ for each $B \subseteq Y$. For each $y \in cl(B)$ denote by j_y an index $j_y \in \mathbb{N}_0$ such that $y \in B_{j_y}$, where B_{j_y} is defined by (8). We have $cl(B) = \bigcup_{y \in cl(B)} B_{j_y}$. Since Y is finite, $cl(B)$ is finite, i.e. for an index $j = \max\{j_y \mid y \in cl(B)\}$, we have $cl(B) = B_j$, where B_j is defined by (8). Therefore, $C((cl(B))^{\downarrow\uparrow}) = C({B_j}^{\downarrow\uparrow}) = B_{j+1} \subseteq cl(B)$, i.e. cl is idempotent. Altogether, cl is a closure operator.

We now prove that $\mathrm{fix}(cl) = \mathrm{Int}_C(X, Y, I)$.

"$\subseteq$": Let $B \in \mathrm{fix}(cl)$, i.e. $B = cl(B)$. Using the aforementioned idea, $cl(B) = B_j$ for some index $j \in J$. Therefore, $B = B_j = C({B_{j-1}}^{\downarrow\uparrow})$ for some $j \in J$ which proves that B is a set of C-attributes. Moreover, $B^{\downarrow\uparrow} = {B_j}^{\downarrow\uparrow} \subseteq C({B_j}^{\downarrow\uparrow}) = B_{j+1} \subseteq cl(B) = B$, i.e. B is an intent of a formal concept in $\langle X, Y, I\rangle$. Putting it together, $B \in \mathrm{Int}_C(X, Y, I)$.

"$\supseteq$": Let $B \in \mathrm{Int}_C(X, Y, I)$. By definition, $B = C(B)$ and $B = B^{\downarrow\uparrow}$. Thus, for each $j \in \mathbb{N}_0$, $B_j = B$, yielding $B = cl(B)$, i.e., $B \in \mathrm{fix}(cl)$. □

Theorem 4 provides a way to compute C-interesting intents and thus the complete lattice of C-concepts: one can use arbitrary algorithm for computing all fixed points of a closure operator because C-interesting intents are exactly the fixed points of a closure operator cl derived from C and ${}^{\downarrow\uparrow}$. Note that

Algorithm 1: Procedure C-INTENTS(B, y)

```
store B;
if B = Y or y > n then
    return
end
for j from y upto n do
    if j ∉ B then
        set D to cl(B ∪ {j});
        if B ∩ Y_j = D ∩ Y_j then
            C-INTENTS(D, j + 1);
        end
    end
end
return
```

well-known algorithms for computing fixed points of closure operators include Ganter's algorithm [8] (algorithm for listing fixed points in a lexical order), Lindig's algorithm [14] (algorithm for listing fixed points and computing their hierarchy), and variants of the algorithm proposed independently by Kuznetsov and Norris [12, 18]. A survey of algorithms for FCA can be found in [13]. For the purpose of illustration, we use an algorithm based on CbO [12].

Suppose that Y contains attributes labeled by consecutive nonnegative integers, i.e., $Y = \{0, 1, \ldots, n\}$. For each $j \in \mathbb{N}_0$, denote by Y_j a subset of attributes defined by

$$Y_j = \{y \in Y \mid y < j\}. \tag{10}$$

Then, an algorithm for computing C-intents can be formalized by a procedure C-INTENTS(B, y) in Algorithm 1. The procedure accepts a C-intent B and an attribute y and recursively lists all C-INTENTS beginning with B. It can be shown that C-INTENTS$(cl(\emptyset), 0)$ lists all C-intents, each of them exactly once. This can be shown in much the same way as in case of the original CbO [12], details are therefore omitted.

Remark 5. Notice that C-INTENTS is a polynomial time-delay algorithm [11] provided that $C(B)$ ($B \subseteq Y$) can be computed with a polynomial time complexity. Indeed, for each $B \subseteq Y$, $B^{\downarrow\uparrow}$ can be computed in a polynomial time (well-known fact). Since Y is finite, there is an index $i \leq |Y|$ such that $cl(B) = B_i$, where B_i is defined by (8). Thus, if $C(B)$ can be computed in a polynomial time, C-INTENTS computes with a polynomial time delay. ■

5. Constraints and Attribute Implications

In this section, we focus on a new type of attribute dependencies which arise in FCA when constraints by closure operators are imposed on attribute

implications. In particular, we deal with validity of the dependencies, non-redundant sets of these dependencies, algorithms to compute the non-redundant sets, and show their relationship to lattices of C-concepts from Section 3.

Throughout this section, we consider a fixed set Y of attributes, a formal context $\langle X, Y, I \rangle$, and a closure operator $C\colon 2^Y \to 2^Y$ representing a constraint. In addition, all attribute implications considered in this section are attribute implications over Y. Dependencies between C-interesting sets of attributes are formalized as follows:

Definition 3. An attribute implication $A \Rightarrow B$ is called a *C-attribute implication* (shortly, a *C-implication*) if A and B are sets of C-attributes.

According to Definition 3, C-implications are but ordinary attribute implications in which both the antecedent and consequent are C-interesting sets of attributes. Validity of C-implications may be defined as validity of ordinary attribute implications. Then, however, a C-implication $A \Rightarrow B$ may be ruled out non-valid because of a non-interesting set of attributes (a set M of attributes of an object for which $A \subseteq M$ and $B \not\subseteq M$). This is not desirable because only C-interesting sets of attributes should be regarded as reasons to reject a C-implication. Therefore, we propose the following approach in which a C-implication is checked in all C-interesting sets of attributes shared by objects from X.

Definition 4. A C-implication $A \Rightarrow B$ is called *C-valid in a formal context* $\langle X, Y, I \rangle$ if, for any $N \subseteq X$, the following condition is satisfied:

$$\text{if } N^{\uparrow} \text{ is a set of } C\text{-attributes, then } N^{\uparrow} \models A \Rightarrow B.$$

The fact that $A \Rightarrow B$ is C-valid in $\langle X, Y, I \rangle$ is denoted by $I \models_C A \Rightarrow B$.

Notice that if C is the identity operator (i.e. C does not impose any constraint), C-validity coincides with the ordinary notion of validity of attribute implications. Testing of C-validity for given C-implication in a given formal context $\langle X, Y, I \rangle$ according to Definition 4 is more complex than testing validity of attribute implications. Indeed, unlike the ordinary case, where we perform a single test for each object, according to Definition 4, we have to perform a test for each subset of objects. Thus, the number of tests grows exponentially with the size of X. It is therefore important to have a quick test for checking $I \models_C A \Rightarrow B$. The following assertion shows several equivalent criteria for C-validity which can be used for quick tests.

Theorem 5. *Let $\langle X, Y, I \rangle$ be a formal context and let $A \Rightarrow B$ be a C-implication. The following are equivalent:*

*(*i*)* $I \models_C A \Rightarrow B$,

*(*ii*)* $\mathrm{Int}_C(X, Y, I) \models A \Rightarrow B$,

*(*iii*)* $cl(A) \models A \Rightarrow B$,

*(*iv*)* $B \subseteq cl(A)$,

where $cl\colon 2^Y \to 2^Y$ *is defined by* (9).

Proof. "$(i) \Rightarrow (ii)$": Let $I \models_C A \Rightarrow B$. For any $M \in \mathrm{Int}_C(X,Y,I)$, $\langle M^{\downarrow}, M\rangle$ is C-concept because $M^{\downarrow\uparrow} = M$. Put $N = M^{\downarrow} \subseteq X$. Using $I \models_C A \Rightarrow B$, we get $N^{\uparrow} \models A \Rightarrow B$, i.e., $M \models A \Rightarrow B$ because $N^{\uparrow} = M^{\downarrow\uparrow} = M$ is a set of C-attributes, see Definition 4.

"$(ii) \Rightarrow (iii)$": Trivial because $cl(A) \in \mathrm{Int}_C(X,Y,I)$.

"$(iii) \Rightarrow (iv)$": Since $A \subseteq cl(A)$, from $cl(A) \models A \Rightarrow B$ it readily follows that $B \subseteq cl(A)$.

"$(iv) \Rightarrow (i)$": Let $B \subseteq cl(A)$ and consider $N \subseteq X$ such that $N^{\uparrow}$ is a set of C-attributes. Then, $N^{\uparrow}$ is a C-intent of a C-concept $\langle N^{\uparrow\downarrow}, N^{\uparrow}\rangle$. Hence, $cl(N^{\uparrow}) = N^{\uparrow}$. Furthermore, if $A \subseteq N^{\uparrow}$, the monotony of cl yields $cl(A) \subseteq cl(N^{\uparrow}) = N^{\uparrow}$. Therefore, $B \subseteq cl(A)$ and the latter inclusion yield $B \subseteq N^{\uparrow}$. Hence, $A \subseteq N^{\uparrow}$ implies $B \subseteq N^{\uparrow}$, proving $N^{\uparrow} \models A \Rightarrow B$. □

Remark 6. (1) Theorem 5 (*ii*) says that $A \Rightarrow B$ is C-valid in $\langle X,Y,I\rangle$ iff it is valid in every C-intent of $\langle X,Y,I\rangle$, generalizing the well-known fact that an implication is valid in $\langle X,Y,I\rangle$ if and only if it is valid in every intent of $\langle X,Y,I\rangle$.

(2) Theorem 5 (*iii*) and (*iv*) provide simple tests of C-validity. Namely, if $C(B)$ ($B \subseteq Y$) can be computed in a polynomial time (which is indeed so in many important cases, cf. Section 7 with examples) then C-validity can be checked in polynomial time as well.

In order to describe bases of C-implications, we need the following notions:

Definition 5. Let T be a set of C-implications. A set $M \subseteq Y$ is called a *C-model of T* if M is a set of C-attributes such that $M \in \mathrm{Mod}(T)$. Denote by $\mathrm{Mod}_C(T)$ the system of all C-models of T, i.e. $\mathrm{Mod}_C(T) = \{M \in \mathrm{Mod}(T) \mid M = C(M)\}$.

Furthermore, a C-implication $A \Rightarrow B$ is *C-entailed by T*, denoted $T \models_C A \Rightarrow B$, if $\mathrm{Mod}_C(T) \models A \Rightarrow B$.

The notions of a completeness and a base are defined as follows:

Definition 6. A set T of C-implications is called *C-complete in $\langle X,Y,I\rangle$* if, for each C-implication $A \Rightarrow B$:

$$T \models_C A \Rightarrow B \text{ iff } I \models_C A \Rightarrow B.$$

Furthermore, T is called a *C-base of $\langle X,Y,I\rangle$* if T is C-complete in $\langle X,Y,I\rangle$ and no proper subset of T is C-complete in $\langle X,Y,I\rangle$.

C-complete sets in $\langle X,Y,I\rangle$ can be characterized as sets of C-implications whose models are exactly C-intents:

Theorem 6. *A set T of C-implications is C-complete in $\langle X,Y,I\rangle$ iff* $\mathrm{Mod}_C(T) = \mathrm{Int}_C(X,Y,I)$.

Proof. Let T be C-complete in $\langle X, Y, I\rangle$. We are going to check that $\mathrm{Mod}_C(T)$ equals $\mathrm{Int}_C(X, Y, I)$ by checking both inclusions.

"$\subseteq$": Let $M \in \mathrm{Mod}_C(T)$. First, observe that $\mathrm{Int}_C(X, Y, I) \models M \Rightarrow cl(M)$. Indeed, for each C-intent $N \in \mathrm{Int}_C(X, Y, I)$, if $M \subseteq N$ then $cl(M) \subseteq cl(N) = N$. Since T is complete, using Theorem 5 (ii), from $\mathrm{Int}_C(X, Y, I) \models M \Rightarrow cl(M)$ it follows that $T \models_C M \Rightarrow cl(M)$. Since $M \in \mathrm{Mod}_C(T)$, we get $M \models M \Rightarrow cl(M)$. Hence, $cl(M) \subseteq M$, showing $M = cl(M)$, i.e., $M \in \mathrm{Int}_C(X, Y, I)$.

"$\supseteq$": Let $M \in \mathrm{Int}_C(X, Y, I)$ and consider any $A \Rightarrow B \in T$. Then, trivially, $T \models_C A \Rightarrow B$. Since T is C-complete, we obtain $I \models_C A \Rightarrow B$. Furthermore, M is a C-intent, i.e., using Theorem 5 (ii), the latter observation yields $M \models A \Rightarrow B$. Since $A \Rightarrow B \in T$ has been taken arbitrarily, we get $M \models A \Rightarrow B$ for all $A \Rightarrow B \in T$, i.e., $M \in \mathrm{Mod}_C(T)$ because $M = C(M)$.

Conversely, if $\mathrm{Mod}_C(T) = \mathrm{Int}_C(X, Y, I)$ then T is obviously C-complete in $\langle X, Y, I\rangle$, see Theorem 5 (ii). □

Remark 7. (1) According to Theorem 6, a set T of C-implications is C-complete if the C-models of T are exactly the C-interesting intents. From this point of view, a C-complete set fully describes the lattice of C-concepts using the notion of a C-model. Consequently, a C-base is a set T of C-implications fully describing C-concepts so that one cannot remove any C-implication from T without losing C-completeness. Hence, C-bases are minimal C-complete sets of C-implications.

(2) In general, a C-complete set T of attribute implications (C-implications) may have models which are not C-models. Also note that if C is given by $C(B) = B$ ($B \in 2^Y$), then the notions of a C-model and a C-completeness coincide with the ordinary notions of a model and a completeness [9].

The following theorem shows that bases are complete sets which are not redundant in the sense that no C-implication from a base follows from the other C-implications:

Theorem 7. *Let T be C-complete in $\langle X, Y, I\rangle$. Then, T is a C-base of $\langle X, Y, I\rangle$ iff $T - \{A \Rightarrow B\} \not\models_C A \Rightarrow B$ holds for each $A \Rightarrow B \in T$.*

Proof. Suppose that T is a C-base of $\langle X, Y, I\rangle$ and consider any $A \Rightarrow B \in T$. Then, $T - \{A \Rightarrow B\}$ is not C-complete and thus $\mathrm{Mod}_C(T) \subset \mathrm{Mod}_C(T - \{A \Rightarrow B\})$, see Theorem 6. In other words, there is $M \in \mathrm{Mod}_C(T - \{A \Rightarrow B\})$ which is not a C-model of T. Therefore, $M \not\models A \Rightarrow B$. Hence, $T - \{A \Rightarrow B\} \not\models_C A \Rightarrow B$.

Conversely, let $T - \{A \Rightarrow B\} \not\models_C A \Rightarrow B$ for each $A \Rightarrow B \in T$ and take $T' \subset T$. Then, there is $A \Rightarrow B \in T$ such that $A \Rightarrow B \notin T'$. Thus, $T \models_C A \Rightarrow B$ and $T' \not\models_C A \Rightarrow B$, i.e., T' is not C-complete in $\langle X, Y, I\rangle$. As a consequence, T is a base of $\langle X, Y, I\rangle$. □

We now show a way to find particular C-bases. For this purpose, we introduce the following notion, generalizing the classic notion of a pseudo-intent [9, 10]:

Definition 7. A set P of C-attributes is called a *C-pseudo-intent of formal context* $\langle X, Y, I\rangle$ if $P \subset cl(P)$ and, for each C-pseudo-intent Q of $\langle X, Y, I\rangle$ such that $Q \subset P$, we have $cl(Q) \subseteq P$. The set of all C-pseudo-intents of $\langle X, Y, I\rangle$ is denoted by $\mathcal{P}_C(X, Y, I)$.

Remark 8. (1) Notice that Definition 7 is a recursive definition in which C-pseudo-intents are defined by means of strictly smaller C-pseudo-intents. Also note that each C-pseudo-intent is a set of C-attributes which is not a C-intent. If $C(\emptyset)$ is closed under $^{\downarrow\uparrow}$, then $C(\emptyset)$ is the least C-intent; otherwise $C(\emptyset)$ is the least C-pseudo-intent.

(2) If C is the identity map, the notion of a C-pseudo-intent coincides with the notion of a pseudo-intent, see [9, 10].

C-pseudo-intents are important because they determine an important C-base. Namely, given $\mathcal{P}_C(X, Y, I)$, one can form a set T of C-implications

$$T = \{P \Rightarrow cl(P) \,|\, P \in \mathcal{P}_C(X, Y, I)\}. \tag{11}$$

For T, we can prove the following:

Theorem 8. *T defined by* (11) *is a C-base of* $\langle X, Y, I\rangle$.

Proof. Notice that T is obviously a set of C-implications. We now check that T is C-complete. Due to Theorem 6, it suffices to check that $\mathrm{Mod}_C(T) = \mathrm{Int}_C(X, Y, I)$.

"$\subseteq$": Let $M \in \mathrm{Mod}_C(T)$. Thus, M is a set of C-attributes. By contradiction, let $M \neq cl(M)$, i.e. $M \subset cl(M)$ because cl is extensive. Now, for each C-pseudo-intent Q, we have $M \models Q \Rightarrow cl(Q)$ because M is a model of T. Therefore, for each C-pseudo-intent Q, if $Q \subset M$ then $cl(Q) \subseteq M$, i.e. M is a C-pseudo-intent by Definition 7. On the other hand, $M \not\models M \Rightarrow cl(M)$ because $cl(M) \not\subseteq M$, a contradiction to $M \in \mathrm{Mod}_C(T)$.

"$\supseteq$": Let $M \in \mathrm{Int}_C(X, Y, I)$. Then $M = cl(M)$. For each C-pseudo-intent P, if $P \subseteq M$ then $cl(P) \subseteq cl(M) = M$, i.e. $M \models P \Rightarrow cl(P)$. Altogether, $M \in \mathrm{Mod}_C(T)$.

It remains to check that T is a C-base. For each C-pseudo-intent P, $P \models Q \Rightarrow cl(Q)$ where $Q \neq P$ is any C-pseudo-intent. Thus, P is a C-model of $T_P = T - \{P \Rightarrow cl(P)\}$ which gives $\mathrm{Mod}_C(T_P) \supset \mathrm{Int}_C(X, Y, I)$, i.e. T_P is not C-complete in $\langle X, Y, I\rangle$. □

Due to Theorem 8, in order to get a C-base, it suffices to compute all C-pseudo-intents. To do that, we express C-pseudo-intents as particular fixed points of a newly constructed closure operator. Given a set T of C-implications, we consider sets B_i defined by $B_0 = B$ and

$$B_{i+1} = C\big(B_i \cup \bigcup\{D \,|\, A \Rightarrow D \in T \text{ and } A \subset B_i\}\big),$$

for each integer $i \geq 0$. Obviously $B_i \subseteq B_{i+1}$ for each $i \geq 0$. Furthermore, we define an operator $cl_T \colon 2^Y \to 2^Y$ as follows:

$$cl_T(B) = \textstyle\bigcup_{i=0}^{\infty} B_i\,. \tag{12}$$

Algorithm 2: Procedure C-BASE($\langle X, Y, I \rangle$)

```
set T to ∅;
set B to C(∅);
while B ≠ Y do
    if B ≠ cl(B) then
        add B ⇒ cl(B) to T;
    end
    set B to NextClosure(B, cl_T)
end
return T
```

We now have the following

Theorem 9. *Let T be defined by* (11)*. Then cl_T defined by* (12) *is a closure operator such that*

$$\mathrm{fix}(cl_T) = \mathcal{P}_C(X, Y, I) \cup \mathrm{Int}_C(X, Y, I).$$

Proof. cl_T is indeed a closure operator (apply arguments from the proof of Theorem 4). We check that $\mathrm{fix}(cl_T) = \mathcal{P}_C(X, Y, I) \cup \mathrm{Int}_C(X, Y, I)$.

"$\subseteq$": Let $B = cl_T(B)$. If $B \notin \mathrm{Int}_C(X, Y, I)$, it suffices to check that B is a C-pseudo-intent. Since Y is finite, $B = cl_T(B) = B_{i_0}$ for some $i_0 \geq 1$. That is, B is of the form $C(\cdots)$, yielding that B is a set of C-attributes. Moreover, for each C-pseudo-intent Q, if $Q \subset B$ then $cl(Q) \subseteq B$ because $B = cl_T(B) = B_{i_0} = B_{i_0+1}$ and $Q \Rightarrow cl(Q) \in T$. Therefore, B is a C-pseudo-intent.

"$\supseteq$": Clearly, for each C-intent B, $B_i = B$ ($i \in \mathbb{N}$), i.e. B is a fixed point of cl_T. The same is true if B is a C-pseudo-intent. □

Theorem 9 says that the fixed points of cl_T are all the C-pseudo-intents together with all the C-intents. This provides us with a way to determine a C-base: we can use the NextClosure [9] algorithm to compute $\mathrm{fix}(cl_T)$ and then single out the C-pseudo-intents according to $\mathcal{P}_C(X, Y, I) = \{P \in \mathrm{fix}(cl_T) \mid P \neq cl(P)\}$. That is, the C-base (11) becomes

$$T = \{P \Rightarrow cl(P) \mid P = cl_T(P) \text{ and } P \neq cl(P)\}$$

due to Theorem 8. The algorithm is described in Algorithm 2. The procedure accepts a formal context as the input and produces a C-base (11). The procedure utilizes procedure NextClosure which, for given set B of attributes and closure operator cl_T computes the least lexical successor of B, see [8, 9] for details. This ensures that T is properly updated and during each step of the computation, all C-implications which are needed to determine next C-pseudo-intent are already present in T. This is easily seen since the lexical order is a total strict order on 2^Y extending the proper subsethood "$\subset$".

6. Syntactic Entailment and Axiomatization

In this section, we investigate syntactic entailment (i.e. provability) of C-implications as a counterpart to the semantic entailment $\models_C$ introduced in Section 5. We provide a simple set of deduction rules, inspired by the well-known Armstrong deduction rules [1, 15], and show that the rules are syntactico-semantically complete in that a given C-implication is semantically entailed by a set T of C-implications if and only if the C-implication if provable from T using these rules.

A version of Armstrong deduction rules, which we use in what follows, consists of the following two rules which describe what attribute implications can be derived (in a single inference step) from other implications:

(Ax): infer $E \cup F \Rightarrow F$,

(Cut): from $E \Rightarrow F$ and $F \cup G \Rightarrow H$ infer $E \cup G \Rightarrow H$.

Remark 9. In literature, several other equivalent sets of deduction rules are described [15]. We have chosen this particular set of rules because it is concise. Also note that (Ax) is, in fact, an axiom scheme stating that each attribute implication of the from $E \cup F \Rightarrow F$ is derivable in a single step (from no preceding attribute implications). (Cut) is a deduction rule which produces an attribute implication $E \cup G \Rightarrow H$ from two attribute implications of the form $E \Rightarrow F$ and $F \cup G \Rightarrow H$.

Using (Ax) and (Cut), one defines the notions of a proof and a provability. A sequence $\varphi_1, \ldots, \varphi_n$ of attribute implications is called a *proof of φ from T* if $\varphi_n = \varphi$ and, for each $i = 1, \ldots, n$, we have

(i) $\varphi_i \in T$ or

(ii) φ_i results using (Ax), or

(iii) φ_i results from φ_j and φ_k $(j, k < i)$ using (Cut).

An attribute implication $A \Rightarrow B$ *is provable from* T, written $T \vdash A \Rightarrow B$, if there is a proof of $A \Rightarrow B$ from T.

Remark 10. In addition to (Ax) and (Cut), one can use so-called derived deduction rules, i.e. rules that can be used as shorthands for certain repeated applications of (Ax) and (Cut). We use the following derived deduction rules:

(Tra): from $E \Rightarrow F$ and $F \Rightarrow H$ infer $E \Rightarrow H$,

(Wea): from $E \Rightarrow F$ infer $E \cup G \Rightarrow F$.

It is easily seen that (Tra) (rule of transitivity) is, in fact, an instance of (Cut) for $G = \emptyset$. Moreover, (Wea) (rule of weakening) is a derived rule because it follows by (Cut) from $E \Rightarrow F$ and $F \cup G \Rightarrow F$, which is an instance of (Ax).

The relationship between the semantic entailment $\models$ and provability $\vdash$ is characterized by the following well-known completeness result [1, 9, 15]:

Theorem 10. *For any set T of attribute implications and any attribute implication $A \Rightarrow B$, we have*

$$T \vdash A \Rightarrow B \quad \textit{iff} \quad T \models A \Rightarrow B,$$

i.e., $A \Rightarrow B$ is entailed by T iff $A \Rightarrow B$ is provable from T.

We now turn our attention to provability of C-implications.

First, we show that the ordinary provability $\vdash$ of attribute implications from a set T of C-implications can be used to characterize C-entailment from T provided that we consider provability form T enriched by additional attribute implications. For any set T of C-implications, consider the set T_C of attribute implications

$$T_C = T \cup \mathrm{Th}(C), \tag{13}$$

where $\mathrm{Th}(C)$ is any set of attribute implications such that

$$\mathrm{Mod}(\mathrm{Th}(C)) = \mathrm{fix}(C). \tag{14}$$

Observe that such $\mathrm{Th}(C)$ always exists. For instance, one may take

$$\mathrm{Th}(C) = \{A \Rightarrow C(A) \,|\, A \subseteq Y\}. \tag{15}$$

Instead of taking this particular $\mathrm{Th}(C)$ which is large since $|\mathrm{Th}(C)| = 2^{|Y|}$, one can take any equivalent non-redundant set of attribute implications, e.g., the Guigues-Duquenne base of all fixed points of C, see [10]. For our argument below, it is important that $\mathrm{Th}(C)$ satisfying (14) always exists.

We now get the following characterization:

Theorem 11. *For any set T of C-implications and any C-implication $A \Rightarrow B$, the following are true*

*(*i*)* $\mathrm{Mod}(T_C) = \mathrm{Mod}_C(T)$,

*(*ii*)* $T_C \models A \Rightarrow B$ *iff* $T \models_C A \Rightarrow B$,

where T_C is given by (13) *with* $\mathrm{Th}(C)$ *satisfying* (14).

Proof. Using (13) and (14), we get

$$\begin{aligned} \mathrm{Mod}(T_C) &= \mathrm{Mod}(T \cup \mathrm{Th}(C)) = \\ &= \mathrm{Mod}(T) \cap \mathrm{Mod}(\mathrm{Th}(C)) = \\ &= \mathrm{Mod}(T) \cap \mathrm{fix}(C) = \mathrm{Mod}_C(T), \end{aligned}$$

proving (i). (ii) follows immediately from (i). □

As a consequence of the previous assertions:

Theorem 12. *For any set T of C-implications and any C-implication $A \Rightarrow B$, we have*

$$T_C \vdash A \Rightarrow B \quad \textit{iff} \quad T \models_C A \Rightarrow B,$$

where T_C is given by (13) *with* $\mathrm{Th}(C)$ *satisfying* (14).

Proof. Consequence of Theorem 10 and Theorem 11. □

Theorem 12 can be seen as a completeness result for C-entailment. However, it uses the ordinary notion of provability of attribute implications. Therefore, corresponding proofs of C-implications from T_C contain attribute implications which are not C-implications. This kind of mixing C-implications and general attribute implications in proofs is undesirable. Therefore, we are interested in finding a system of Armstrong-like deduction rules which enables us to infer C-implications from C-implications, so that the proofs contain only C-implications.

Consider the following deduction rules:

(Ax_C): infer $C(E \cup F) \Rightarrow F$, where $F \in \mathrm{fix}(C)$,

(Cut_C): from C-implications $E \Rightarrow F$ and $C(F \cup G) \Rightarrow H$
infer $C(E \cup G) \Rightarrow H$.

Remark 11. (1) Notice that (Ax_C) and (Cut_C) are indeed rules which produce C-implications from input C-implications.

(2) Deduction rules (Ax_C) and (Cut_C) generalize the original Armstrong rules (Ax) and (Cut) the following way: if C is identity map, then (Ax_C) and (Cut_C) become (Ax) and (Cut), respectively.

In much the same way as in case of (Ax) and (Cut), we can define the notions of a C-proof and a C-provability with (Ax_C) and (Cut_C) used instead of (Ax) and (Cut) and considering only C-implications as formulas: A sequence $\varphi_1, \ldots, \varphi_n$ of C-implications is called a *C-proof of φ from T* if $\varphi_n = \varphi$ and, for each $i = 1, \ldots, n$, we have

(i) $\varphi_i \in T$ or

(ii) φ_i results using (Ax_C), or

(iii) φ_i results from φ_j and φ_k $(j, k < i)$ using (Cut_C).

A C-implication $A \Rightarrow B$ *is C-provable from T*, written $T \vdash_C A \Rightarrow B$, if there is a C-proof of $A \Rightarrow B$ from T.

The following assertions show the relationship between the ordinary provability $\vdash$ and C-provability $\vdash_C$:

Theorem 13. *Let T be a set of C-implications. Then, for every C-implication $A \Rightarrow B$,*

$$\textit{if } T_C \vdash A \Rightarrow B, \textit{ then } T \vdash_C A \Rightarrow B,$$

where T_C is given by (13) *with* $\mathrm{Th}(C)$ *defined by* (15).

Proof. Let $T_C \vdash A \Rightarrow B$. Thus, there is a proof $A_1 \Rightarrow B_1, \ldots, A_n \Rightarrow B_n$ of $A \Rightarrow B$ from T_C. We show, by induction on the length of the proof, that $T \vdash_C C(A_i) \Rightarrow C(B_i)$ for all $i = 1, \ldots, n$. As a particular case, we get $T \vdash_C C(A_n) \Rightarrow C(B_n)$, showing $T \vdash_C A \Rightarrow B$ because $A_n = A = C(A)$ and $B_n = B = C(B)$, respectively.

Take $i \leq n$ and suppose that the claim holds for all $j < i$. If $A_i \Rightarrow B_i \in T$, then obviously $T \vdash_C C(A_i) \Rightarrow C(B_i)$ because $A_i \Rightarrow B_i$ is a C-implication.

If $A_i \Rightarrow B_i \in \mathrm{Th}(C)$, then from (15), $B_i = C(A_i)$, i.e. $A_i \Rightarrow B_i$ can be written as $A_i \Rightarrow C(A_i)$. Observe that $C(A_i) \Rightarrow C(C(A_i))$ becomes $C(A_i) \Rightarrow C(A_i)$ because C is idempotent. Hence, $C(A_i) \Rightarrow C(B_i)$ equals $C(A_i) \Rightarrow C(A_i)$ for which $T \vdash_C C(A_i) \Rightarrow C(A_i)$ using (Ax_C), hence $T \vdash_C C(A_i) \Rightarrow C(B_i)$.

If $A_i \Rightarrow B_i$ is an instance of (Ax), then $B_i \subseteq A_i$. Since C is monotone, we get that $C(B_i) \subseteq C(A_i)$, i.e., $C(A_i) \Rightarrow C(B_i)$ is an instance of (Ax_C) because $C(A_i) \Rightarrow C(B_i)$ equals $C(A_i \cup C(B_i)) \Rightarrow C(B_i)$, showing $T \vdash_C C(A_i) \Rightarrow C(B_i)$.

Finally, let $A_i \Rightarrow B_i$ result from $E \Rightarrow F$ and $F \cup G \Rightarrow H$ by (Cut). In that case, $A_i = E \cup G$ and $B_i = H$. Using the induction hypothesis, $T \vdash_C C(E) \Rightarrow C(F)$ and $T \vdash_C C(F \cup G) \Rightarrow C(H)$. Since $C(F \cup G) \Rightarrow C(H)$ can be written as $C(C(F) \cup G) \Rightarrow C(H)$, (Cut_C) yields $C(C(E) \cup G) \Rightarrow C(H)$ which is equal to $C(E \cup G) \Rightarrow C(H)$, i.e., $C(A_i) \Rightarrow C(B_i)$. Since, $C(A_i) \Rightarrow C(B_i)$ has been inferred using (Cut_C) from C-implications which are C-provable from T, $C(A_i) \Rightarrow C(B_i)$ is also C-provable from T, i.e., $T \vdash_C C(A_i) \Rightarrow C(B_i)$, finishing the proof. □

Theorem 14. *Let T be a set of C-implications. Then, for every C-implication $A \Rightarrow B$,*

$$\textit{if } T \vdash_C A \Rightarrow B, \textit{ then } T_C \vdash A \Rightarrow B,$$

where T_C is given by (13) *with* $\mathrm{Th}(C)$ *defined by* (15).

Proof. Let $T \vdash_C A \Rightarrow B$. Thus, there is a C-proof $A_1 \Rightarrow B_1, \ldots, A_n \Rightarrow B_n$ of $A \Rightarrow B$ from T. We show, by induction on the length of the proof, that $T_C \vdash A_i \Rightarrow B_i$ for all $i = 1, \ldots, n$.

Take $i \leq n$ and suppose that the claim holds for all $j < i$. If $A_i \Rightarrow B_i \in T$ then trivially $T_C \vdash A_i \Rightarrow B_i$ because $T \subseteq T_C$. If $A_i \Rightarrow B_i$ is an instance of (Ax_C), then $A_i = C(E \cup B_i)$ for some $E \subseteq Y$. In this case, $T_C \vdash C(E \cup B_i) \Rightarrow B_i$, because $C(E \cup B_i) \Rightarrow B_i$ is an instance of (Ax). Indeed, this is a consequence of extensivity of C because $B_i \subseteq E \cup B_i \subseteq C(E \cup B_i)$. Let $A_i \Rightarrow B_i$ result from C-implications $E \Rightarrow F$ and $C(F \cup G) \Rightarrow H$ by (Cut_C). Then, $A_i = C(E \cup G)$ and $B_i = H$. Using the induction hypothesis, $T_C \vdash E \Rightarrow F$ and $T_C \vdash C(F \cup G) \Rightarrow H$. In addition to that, $T_C \vdash F \cup G \Rightarrow C(F \cup G)$ because $\mathrm{Th}(C) \subseteq T_C$. Thus, using (Tra), which is a derived deduction rule, see Remark 10, it follows that $T_C \vdash F \cup G \Rightarrow H$. Now, using (Cut), we get $T_C \vdash E \cup G \Rightarrow H$. Using the derived deduction rule (Wea) on $E \cup G \Rightarrow H$ and the fact that $E \cup G \subseteq C(E \cup G)$, we get $T_C \vdash C(E \cup G) \Rightarrow H$ which means $T_C \vdash A_i \Rightarrow B_i$, finishing the proof. □

As a consequence:

Theorem 15. *Let T be a set of C-implications. Then, for every C-implication $A \Rightarrow B$, we have*

$$T_C \vdash A \Rightarrow B \quad \textit{iff} \quad T \vdash_C A \Rightarrow B,$$

where T_C is given by (13) *with* $\mathrm{Th}(C)$ *defined by* (15).

Proof. Consequence of Theorem 13 and Theorem 14. □

We now obtain a completeness theorem that characterizes entailment of C-implications:

Theorem 16 (completeness)**.** *Let T be a set of C-implications. Then, for every C-implication $A \Rightarrow B$, we have*

$$T \vdash_C A \Rightarrow B \quad \textit{iff} \quad T \models_C A \Rightarrow B.$$

Proof. Consequence of Theorem 12 and Theorem 15. □

7. Examples of Constraints

In this section, we provide several examples of constraints that can be defined in terms of closure operators. That is, we provide examples of particular constraints that are covered by the general approach to constraints proposed in this paper. For illustration, we use the data from Fig. 1 to show effect of these constraints.

For each constraint introduced, we present a verbal description of its meaning and a definition of the corresponding closure operator C. Since each constraint is parameterized (i.e., selecting a particular parameter, we obtain a particular constraint), we denote each constraint by its name and a set of parameters in the form

"constraint-name(*parameters*)".

In the following examples, we visualize the constrained concept lattices. In view of Theorem 2, we draw $\mathcal{B}_C(X, Y, I)$ into the diagram of $\mathcal{B}(X, Y, I)$ using a special notation: "•" denote concepts of $\mathcal{B}(X, Y, I)$ that are not present in $\mathcal{B}_C(X, Y, I)$; "○" denote C-concepts; dotted lines denote edges of the original concept lattice that are not present in $\mathcal{B}_C(X, Y, I)$; bold solid lines denote edges that are are present in both $\mathcal{B}(X, Y, I)$ and $\mathcal{B}_C(X, Y, I)$; bold dashed lines denote new edges, i.e. those in $\mathcal{B}_C(X, Y, I)$ that are not in $\mathcal{B}(X, Y, I)$.

Required Attributes: having(Z)
For any $Z \subseteq Y$, C defined by $C(B) = B \cup Z$ is a closure operator. C-intents are just the intents that have all attributes from Z. Notice that the boundary cases mentioned in Remark 3 result by taking $Z = \emptyset$ and $Z = Y$, respectively. For instance, having($\{a, d\}$) determines a constraint on "products containing

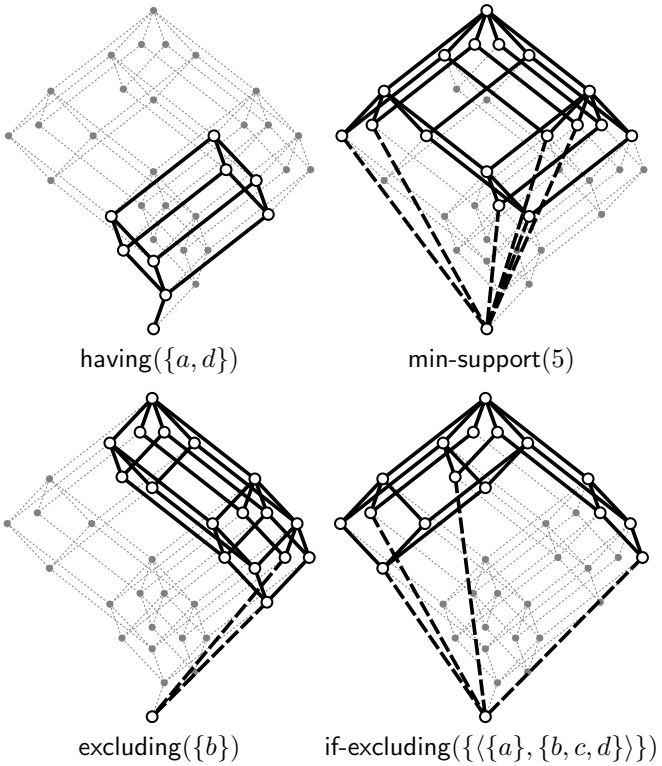

Figure 3: Constrained concept lattices I.

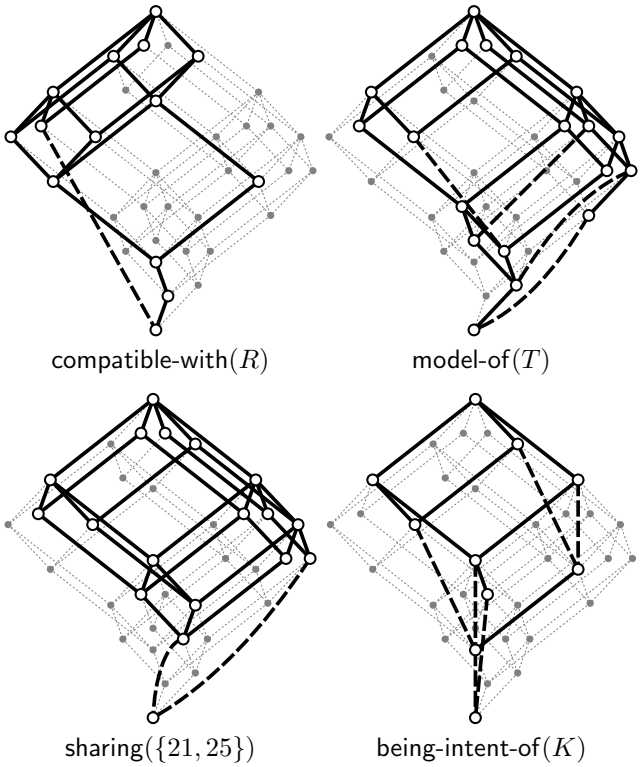

Figure 4: Constrained concept lattices II.

lecithin and mono- and diglycerides", see Fig. 3 (top-left). The C-base for C being $\mathsf{having}(\{a, d\})$ consists of a single C-implication, namely $\{a, c, d\} \Rightarrow Y$.

Maximum Number of Attributes: $\mathsf{max\text{-}attributes}(n)$
In this case, B is considered interesting iff B contains at most n attributes or $B = Y$. Namely, the corresponding closure operator is given by $C(B) = B$ if $|B| \leq n$ and $C(B) = Y$ otherwise.

Required Minimal Support: $\mathsf{min\text{-}support}(s)$
Define C so that B is C-interesting iff $B = Y$ or $|B^{\downarrow}| \geq s$ where s is a non-negative integer. It is easy to see that C-interesting sets form a closure system. $|B^{\downarrow}| \geq s$ means that the number of objects sharing all attributes from B exceeds a user-defined parameter s called *support* in association rules [23]. Condition $B = Y$ is a technical one to ensure that C-interesting sets form a closure system. The corresponding closure operator is defined by

$$C(B) = \begin{cases} B, & \text{if } |B^{\downarrow}| \geq s, \\ Y, & \text{otherwise.} \end{cases}$$

Then, the set $\mathrm{Int}_C(X, Y, I)$ of all C-interesting intents (possibly without Y) coincides with the set of closed frequent itemsets defined by Zaki [22] in order to get non-redundant association rules. The result for $\mathsf{min\text{-}support}(5)$ is depicted in Fig. 3 (top-right). The $\mathsf{min\text{-}support}(5)$-base consists of a single implication $\{b, f\} \Rightarrow \{a, b, f\}$.

Required Minimal Weight: $\mathsf{min\text{-}weight}(m, \lhd, w)$
Constraints by minimal support can be generalized by requiring a minimal weight w of the extent with the weight of extents given by an additional measure-like map m. In general, such m is supposed to be a map $m\colon \mathrm{Ext}(X, Y, I) \to W$ assigning to each extent (or, more generally, to each subset of X) $A \subseteq X$ a weight $m(A)$ from a set W of weights and w is supposed to be a weight from W. In order to compare weights, W is equipped with a transitive relation $\lhd$ satisfying the following monotony condition with respect to m:

$$\text{if } A_1 \subseteq A_2 \text{ then } m(A_1) \lhd m(A_2) \tag{16}$$

for all $A_1, A_2 \in \mathrm{Ext}(X, Y, I)$. Then, $B \subseteq Y$ is C-interesting according to this constraint iff $w \lhd m(B^{\downarrow})$. Hence, the corresponding closure operator is

$$C(B) = \begin{cases} B, & \text{if } w \lhd m(B^{\downarrow}), \\ Y, & \text{otherwise.} \end{cases}$$

C is indeed a closure operator. Namely, C is obviously extensive and idempotent. In order to check monotony of C, we verify a stronger condition, namely that $C(B_2) = B_2$ yields $C(B_1) = B_1$ for all $B_1 \subseteq B_2 \subset Y$. Thus, let $C(B_2) = B_2$ and $B_1 \subseteq B_2 \subset Y$. Since $C(B_2) = B_2 \subset Y$, the definition of C gives $w \lhd m(B_2^{\downarrow})$. Furthermore, from $B_1 \subseteq B_2$, it follows that $B_2^{\downarrow} \subseteq B_1^{\downarrow}$. Using (16), one gets $m(B_2^{\downarrow}) \lhd m(B_1^{\downarrow})$. Transitivity of $\lhd$ then yields $w \lhd m(B_1^{\downarrow})$, i.e., $C(B_1) = B_1$. Hence, C is monotone. Putting it together, C is a closure operator.

Remark 12. (1) Since the values of W are interpreted as weights, it is natural to consider (16) because larger extents are expected to be heavier. Transitivity of $\lhd$ corresponds to the idea that if $A \subseteq X$ is "heavy enough to satisfy the constraint", then any concept with its extent heavier than A should also satisfy the constraint.

(2) It is easy to see that $\mathsf{min\text{-}weight}(m, \lhd, w)$ is a generalization of $\mathsf{min\text{-}support}(s)$ in which all objects contribute to the weigh of an extent equally. Indeed, $\mathsf{min\text{-}support}(s)$ can be seen as $\mathsf{min\text{-}weight}(m, \lhd, s)$ for m defined by $m(A) = |A|$ and considering the genuine ordering $\leq$ of $W = \{0, 1, \dots\}$ as the relation $\lhd$.

(3) Let $W = [0, \infty)$ be a set of non-negative real numbers and let m be a super-additive function, i.e., let m satisfy

$$m(A_1 \cup A_2) \geq m(A_1) + m(A_2) \tag{17}$$

for all sets $A_1, A_2 \subseteq [0, \infty)$ such that $A_1 \cap A_2 = \emptyset$. Then, $\mathsf{min\text{-}weight}(m, \leq, w)$ is a well-defined constraint. Indeed, it suffices to check (16). If $A_1 \subseteq A_2$, using super-additivity, we get

$$m(A_2) = m(A_1 \cup (A_2 - A_1)) \geq m(A_1) + m(A_2 - A_1)$$

Since $m(A_2 - A_1) \geq 0$, we get $m(A_1) \leq m(A_2)$. As a consequence, $\mathsf{min\text{-}weight}(m, \leq, w)$ is well defined.

(4) A particular case of the constraint described in (3) is a constraint by a measure m, i.e., by a non-negative additive function m such that $m(\emptyset) = 0$. In particular, m can be a discrete probability measure $m : 2^X \to [0, 1]$ which is uniquely determined by the values of $m(\{x\}) \in [0, 1]$ for all $x \in X$. This can be interpreted so that each object in the context is assigned its weight which is interpreted as a probability that particular object will be observed (i.e., objects can be seen as elementary events). Under this assumption, extents of concepts are events and the constraint $\mathsf{min\text{-}weight}(m, \leq, w)$ says that a concept $\langle A, B \rangle$ is interesting iff the probability that A occurs is greater than or equal to w.

Example 4. Consider the data $\langle X, Y, I \rangle$ from Fig. 1 and constraints $\mathsf{min\text{-}weight}(m_1, \leq, 7)$ and $\mathsf{min\text{-}weight}(m_2, \leq, 7)$ with functions m_1 and m_2 defined by $m_1(A) = \sum_{x \in A} w_1(x)$ and $m_2(A) = \sum_{x \in A} w_2(x)$, where

$$w_1(x) = \begin{cases} 3, & \text{if } x \in \{14, 15, 17, 18, 19\}, \\ 1, & \text{otherwise,} \end{cases}$$

$$w_2(x) = \begin{cases} 4, & \text{if } x \in \{5, 16, 26\}, \\ 1, & \text{otherwise.} \end{cases}$$

The results are depicted in Fig. 5. The $\mathsf{min\text{-}weight}(m_1, \leq, 7)$-base consists of a single implication $\{b, f\} \Rightarrow \{a, b, f\}$, the $\mathsf{min\text{-}weight}(m_2, \leq, 7)$-base consists of $\{e, f\} \Rightarrow \{a, e, f\}$.

Remark 13. (1) Notice that a constraint analogous to $\mathsf{min\text{-}weight}(m, \lhd, w)$ can be formed in terms of the intents rather than the extents. Namely, one can

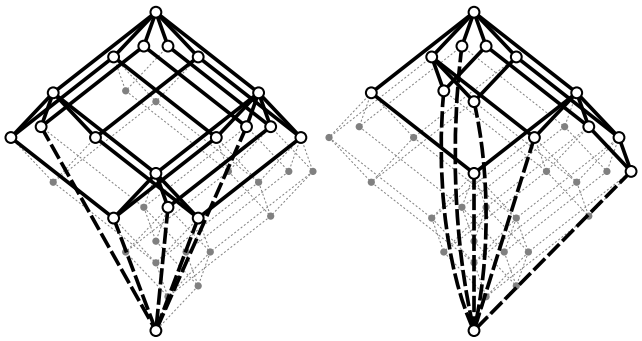

Figure 5: Constrained concept lattices III.

consider max-weight$(m, \lhd, w)$ where $m : \mathrm{Int}(X, Y, I) \to W$ satisfies (16) with respect to a transitive relation $\lhd$ defined on W. Under these conditions, $B \subseteq Y$ is C-interesting according to this constraint iff $m(B) \lhd w$, i.e., if the weight of B is at most w. Obviously, the corresponding closure operator is

$$C(B) = \begin{cases} B, & \text{if } m(B) \lhd w, \\ Y, & \text{otherwise,} \end{cases}$$

and C is a closure operator. Again, as a special case, we can consider max-weight$(m, \lhd, w)$ as a constraint by prescribing probabilities for attributes, see Remark 12 (4).

(2) Obviously, max-weight$(m, \leq, n)$ with $m(B) = |B|$ (a special case of the constraint discussed in (1)) is a generalization of max-attributes(n).

Excluded Attributes: excluding(Z)
For any $Z \subseteq Y$, let B be C-interesting if and only if B does not contain any attribute from Z (or $B = Y$). That is, put

$$C(B) = \begin{cases} B, & \text{if } B \cap Z = \emptyset, \\ Y, & \text{otherwise.} \end{cases}$$

Fig. 3 (bottom-left) contains a result of applying constraint excluding$(\{b\})$ of products not containing the citric acid. In this particular case, the excluding$(\{b\})$-base consists of four implications:

$$\{e, f\} \Rightarrow \{a, e, f\}, \qquad \{c, f\} \Rightarrow Y,$$
$$\{c, e\} \Rightarrow \{a, c, e\}, \qquad \{c, d\} \Rightarrow Y.$$

Conditionally Excluded Attributes: if-excluding(E)
The previous constraint can be generalized so that some attributes are required to be absent provided some other attributes are present. Namely, given a set $E \subseteq 2^Y \times 2^Y$, each $\langle Z_1, Z_2 \rangle \in E$ can be seen as a rule saying "if all attributes

from Z_1 are present then no attribute from Z_2 is present", with a possible exception of set Y.

The corresponding operator is defined as follows: $C(B) = B$ if, for each $\langle Z_1, Z_2\rangle \in E$, either $Z_1 \not\subseteq B$ or $Z_2 \cap B = \emptyset$; otherwise, $C(B) = Y$. C is a closure operator. Indeed, extensivity and idempotency are immediate. Monotony: if $B_1 \subseteq B_2$ and if $Z_1 \subseteq B_1$ and $Z_2 \cap B_1 \neq \emptyset$ then $Z_1 \subseteq B_2$ and $Z_2 \cap B_2 \neq \emptyset$, i.e. $C(B_1) = Y = C(B_2)$. The rest is easy to see. An example of if-excluding(Z) for $Z = \{\langle\{a\}, \{b, c, d\}\rangle\}$ saying that "if a product contains lecithin then it contains neither of citric acid, pectin, and mono- and diglycerides" and the corresponding constrained concept lattice is shown in Fig. 3 (bottom-right). The excluding(Z)-base is the following:

$$\{e, f\} \Rightarrow \{a, e, f\}, \qquad \{c, f\} \Rightarrow Y, \qquad \{c, e\} \Rightarrow Y,$$
$$\{c, d\} \Rightarrow Y, \qquad \{b, f\} \Rightarrow Y.$$

Notice that constraint excluding(Z) is the same constraint as if-excluding($\{\langle\emptyset, Z\rangle\}$).

Compatibility with Relation: compatible-with(R)
This constraint is defined by requiring that interesting set of attributes be compatible with a given binary relation. Consider a binary relation $R \subseteq Y \times Y$. Put

$$B_j = \begin{cases} B, & \text{if } j = 0, \\ B_{j-1} \cup \{y' \mid \text{there is } y \in B_{j-1}\colon \langle y, y'\rangle \in R\}, & \text{if } j \geq 1, \end{cases}$$

and define C by $C(B) = \bigcup_{j=0}^{\infty} B_j$. Since Y is finite, we have $C(B) = B_{j_0}$ for some $j_0 \in \mathbb{N}_0$. Thus, B is C-interesting iff it satisfies the following condition:

$$\text{if } y_1 \in B \text{ and } \langle y_1, y_2\rangle \in R \text{ then } y_2 \in B.$$

For instance, if R is an equivalence relation (i.e., R is reflexive, symmetric, and transitive), see [2],

$$C(B) = \bigcup\{[y]_R \mid y \in B\},$$

where $[y]_R$ denotes the class of R containing y. Fig. 4 (top-left) shows compatible-with($\{\langle a, d\rangle, \langle a, e\rangle, \langle f, b\rangle\}$). The corresponding base contains tree implications:

$$\{c, e\} \Rightarrow Y, \qquad \{c, d\} \Rightarrow Y, \qquad \{b, f\} \Rightarrow \{a, b, d, e, f\}.$$

Required Attribute Implications: model-of(T)
In this constraint, C-interesting sets are models of (i.e., compatible with) a prescribed set T of attribute implications. Since the system of all models of T is a closure system in Y [9], the corresponding closure operator C can be defined by $C(B) = \bigcup_{j=0}^{\infty} B_j$, where

$$B_j = \begin{cases} B, & \text{if } j = 0, \\ B_{j-1} \cup \bigcup\{D \mid A \Rightarrow D \in T \text{ and } A \subseteq B_{j-1}\}, & \text{if } j \geq 1. \end{cases}$$

One can show that $C(B)$ is the least model of T containing B. Hence, B is C-interesting iff B is a model of attribute implications from T. Notice that this type of definition of a closure operator is, in fact, the most general one, because each closure operator on a finite set of attributes can be completely described by a set of attribute implications. This topic is discussed further in Section 8.

The result of model-of$(\{\{a,b\}\Rightarrow\{c\},\{d\}\Rightarrow\{e,f\}\})$ is depicted in Fig. 4 (top-right). The base consists of four implications:

$$\{e,f\}\Rightarrow\{a,e,f\}, \qquad \{c,f\}\Rightarrow\{a,b,c,f\},$$
$$\{c,e\}\Rightarrow\{a,c,e\}, \qquad \{b,f\}\Rightarrow\{a,b,c,f\}.$$

Constraint by Representative Objects: sharing(W)
Let $W\subseteq X$ be set of objects and let B be C-interesting iff at least one object $x\in W$ has all the attributes from B. Thus, W can be seen as a set of selected "representative objects". The corresponding closure operator is defined by

$$C(B)=\begin{cases} B, & \text{if } W\cap B^{\downarrow}\neq\emptyset,\\ Y, & \text{otherwise.}\end{cases}$$

Described verbally, B is C-interesting iff B equals Y (technical condition) or at least one object in W shares all attributes from B. In other words, the extents of interesting concepts contain at least one object from W. One can easily check that C is indeed a closure operator. The lattice of C-concepts constrained by sharing$(\{21,25\})$ is shown in Fig. 4 (bottom-left). The sharing$(\{21,25\})$-base consists of two implications:

$$\{e,f\}\Rightarrow\{a,e,f\}, \qquad \{c,e\}\Rightarrow\{a,c,e\}.$$

Constraints Learned from Data: being-intent-of(K)
Users often want to analyze a data set $\langle X,Y,I\rangle$ and already have a related data set $\langle X',Y,K\rangle$ with the same set of attributes Y. The additional data set $\langle X',Y,K\rangle$ can be, e.g., data set of sample (interesting) data or a data set of selected artificial objects constructed by an expert. Thus, I represents an input data and K can be used as a background knowledge for I. The user might wish to extract only concepts whose intents are intents of K, i.e., intents of the sample or expert data set. Observe that both ${}^{\downarrow_I\uparrow_I}$ and ${}^{\downarrow_K\uparrow_K}$ are closure operators in Y, i.e., ${}^{\downarrow_K\uparrow_K}$ can be used directly as a constraint.

For example, consider the data table from Fig. 1 and an additional data table $\langle\{x_1,\ldots,x_4\},Y,K\rangle$ containing four representative food products and their attributes:

	a	b	c	d	e	f
x_1	×		×	×	×	
x_2		×	×		×	×
x_3	×	×				×
x_4	×	×		×	×	

If we let $C = {}^{\downarrow\kappa\uparrow\kappa}$ the set $\mathcal{B}_C(X, Y, I)$ of all C-concepts consists of all concepts in $\mathcal{B}(X, Y, I)$ whose intents are also intents of $\mathcal{B}(\{x_1, \ldots, x_4\}, Y, K)$, see Fig. 4 (bottom-right). The corresponding base consists of two implications:

$$\{c, e\} \Rightarrow Y, \qquad \{b, f\} \Rightarrow \{a, b, f\}.$$

Minimal Gap: min-gap(Φ)
The following constraint is inspired by [21]. Let Φ be a positive integer. Call a pair $\langle A, B\rangle$ a Φ-concept of a formal context $\langle X, Y, I\rangle$ if $A = B^{\downarrow}$ and $|B^{\downarrow}| - |(B \cup \{y\})^{\downarrow}| \geq \Phi$ for every $y \notin B$. In [21], the author proposes to consider Φ as a parameter and, given Φ, to extract from the data the collection of all Φ-concepts. Since 1-concepts are exactly formal concepts and since every Φ-concept is also a formal concept, Φ-concepts provide an interesting generalization of formal concepts. In addition, the larger the parameter Φ, the smaller the set of Φ-concepts. Φ may be regarded as a prescribed minimal gap in the size of extents of a subconcept and a superconcept. Now, Φ-concepts are just C-interesting concepts for an appropriate closure operator C. Indeed, let C be defined as follows. Let for $B \subseteq Y$,

$$c(B) = \left\{y \in Y \mid \Phi > |B^{\downarrow}| - |(B \cup \{y\})^{\downarrow}|\right\}$$

and put

$$B_j = \begin{cases} B, & \text{if } j = 0, \\ c(B_{j-1}), & \text{if } j > 0. \end{cases}$$

Finally, let

$$C(B) = \textstyle\bigcup_{j=0}^{\infty} B_j.$$

Then, C is a closure operator and $\langle A, B\rangle$ is a Φ-concept if and only if $B = C(B)$ and $A = B^{\downarrow}$. Hence, $\mathcal{B}_C(X, Y, I)$ is just the set of all Φ-concepts of $\langle X, Y, I\rangle$.

8. Combination of Constraints

Constraints may naturally be combined in a conjunctive manner. That is, for closure operators $C_1, \ldots, C_k$ representing constraints we can consider a closure operator denoted $C_1 \,\&\, \cdots \,\&\, C_k$ whose fixed points are common fixed points of C_is ($i = 1, \ldots, k$). $C_1 \,\&\, \cdots \,\&\, C_k$ represents a conjunctive constraint "C_1 and C_2 and $\cdots$ and C_k".

From the computational point of view, the fixed points of $C_1 \,\&\, \cdots \,\&\, C_k$ can be computed in a similar way as in (9). Namely, for each $B \subseteq Y$ and $j \in \mathbb{N}_0$, put

$$B_j = \begin{cases} B, & \text{if } j = 0, \\ C_1(C_2(\cdots(C_k(B_{j-1}))\cdots)), & \text{if } j \geq 1. \end{cases}$$

Then, one can show that $(C_1 \,\&\, \cdots \,\&\, C_k)(B) = \bigcup_{j=0}^{\infty} B_j$. Since Y is a finite set, for each $B \subseteq Y$ there is $j_0 \in \mathbb{N}_0$ such that $(C_1 \,\&\, \cdots \,\&\, C_k)(B) = B_{j_0}$. For illustration, Fig. 6 (top-left) shows the result of applying a conjunctive constraint

$$\mathsf{sharing}(\{13, 14, 16, 17, 18\}) \,\&\, \mathsf{model\text{-}of}(\{\{a, b\} \Rightarrow \{c\}\}).$$

In addition to conjunctive compositions, one can use conditional constraints representing a composed constraint "if C_1 then C_2". In this case, the situation is more complicated since a conditional composition of two constrains is not a closure-based constraint in general. In order to characterize the constraints, we need a notion of a *filter*: a set $\mathcal{F} \subseteq 2^Y$ is called a filter if (i) for each $B_1, B_2, \in \mathcal{F}$, $B_1 \cap B_2 \in \mathcal{F}$; and (ii) if $B_1 \in \mathcal{F}$ and $B_1 \subseteq B_2$ then $B_2 \in \mathcal{F}$.

Theorem 17. *Let C_1, C_2 be closure operators in Y such that $\mathrm{fix}(C_1)$ is a filter. Then, an operator $C_1 \Rightarrow C_2 \colon 2^Y \to 2^Y$ defined by*

$$(C_1 \Rightarrow C_2)(B) = \begin{cases} C_2(B), & \text{if } B \in \mathrm{fix}(C_1), \\ B, & \text{otherwise} \end{cases} \tag{18}$$

is a closure operator and the following conditions are equivalent:

*(*i*) $B \in \mathrm{fix}(C_1 \Rightarrow C_2)$,*

*(*ii*) if $B \in \mathrm{fix}(C_1)$ then $B \in \mathrm{fix}(C_2)$.*

Proof. Obviously, $C_1 \Rightarrow C_2$ is extensive. It is also idempotent. Indeed, for $B' = (C_1 \Rightarrow C_2)(B)$, two mutually exclusive situations may occur. First, if $B \notin \mathrm{fix}(C_1)$, we have $B' = B$, in which case, $(C_1 \Rightarrow C_2)(B') = B'$ follows directly from (18). Second, if $B \in \mathrm{fix}(C_1)$, we have $B' = C_2(B)$, in which case, $B' \in \mathrm{fix}(C_1)$ because $B \subseteq B'$ and $\mathrm{fix}(C_1)$ is a filter, i.e., $(C_1 \Rightarrow C_2)(B') = C_2(B') = C_2(C_2(B)) = C_2(B) = B'$, showing that $C_1 \Rightarrow C_2$ is idempotent.

To prove monotony, suppose that $B_1 \subseteq B_2$. If $B_1 \in \mathrm{fix}(C_1)$ then $(C_1 \Rightarrow C_2)(B_1) = C_2(B_1)$. Furthermore, the fact that $\mathrm{fix}(C_1)$ is a filter yields $B_2 \in \mathrm{fix}(C_1)$ because $B_1 \subseteq B_2$. Therefore, $(C_1 \Rightarrow C_2)(B_2) = C_2(B_2)$. Thus, the monotony of $C_1 \Rightarrow C_2$ follows from the monotony of C_2.

Observe that $B \in \mathrm{fix}(C_1 \Rightarrow C_2)$ iff either $B \neq C_1(B)$ or $B = C_1(B)$ and $B = C_2(B)$. Thus, $B \in \mathrm{fix}(C_1 \Rightarrow C_2)$ iff $B \in \mathrm{fix}(C_1)$ implies $B \in \mathrm{fix}(C_2)$, finishing the proof. □

Example 5. Fig. 6 (bottom-left) shows a conditional constraint

$$\mathsf{having}(\{b\}) \Rightarrow \mathsf{compatible\text{-}with}(\{\langle a, d\rangle, \langle a, e\rangle, \langle f, b\rangle\}).$$

The difference between conjunctive and conditional constraints is further illustrated in Fig. 6 (right) where we have two constraints $\mathsf{having}(\{a\})$ and $\mathsf{min\text{-}support}(6)$ composed by conjunction (top) and implication (bottom).

Recall that Theorem 17 has a limitation of $\mathrm{fix}(C_1)$ being a filter. We now look at this requirement from the point of view of closure operators which satisfy the requirement:

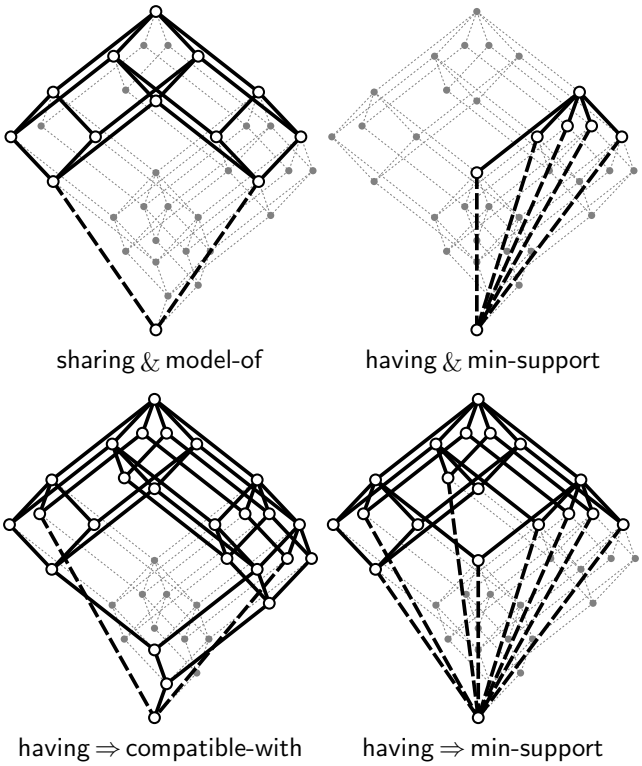

Figure 6: Examples of compound constraints.

Theorem 18. *Let C be a closure operator in Y. Then the following conditions are equivalent:*

*(*i*)* $\mathrm{fix}(C)$ *is a filter,*

*(*ii*)* C *is identical to* $\mathsf{having}(\bigcap \mathrm{fix}(C))$.

Proof. "$(i) \Rightarrow (ii)$": Let $\mathrm{fix}(C)$ be a filter. We prove that $\mathrm{fix}(C)$ equals $\mathrm{fix}(\mathsf{having}(\bigcap \mathrm{fix}(C)))$. According to the definition of $\mathsf{having}(\bigcap \mathrm{fix}(C))$, it suffices to check that $B = C(B)$ iff $\bigcap \mathrm{fix}(C) \subseteq B$. We use the fact that $\bigcap \mathrm{fix}(C)$ is a fixed point of C. Namely, it is the least fixed point of C. So, if $B = C(B)$, then $\bigcap \mathrm{fix}(C) \subseteq B$ because both sets are fixed points of C and $\bigcap \mathrm{fix}(C)$ is the least one. Conversely, if $\bigcap \mathrm{fix}(C) \subseteq B$, the facts that $\mathrm{fix}(C)$ is a filter and that $\bigcap \mathrm{fix}(C)$ is a fixed point of C yield that B is a fixed point of C.

"$(ii) \Rightarrow (i)$" is true because $\mathrm{fix}(\mathsf{having}(Z))$ is a filter for any subset $Z \subseteq Y$, including $Z = \bigcap \mathrm{fix}(C)$. □

Therefore, the limitation of Theorem 17 is quite strong, the only constraints satisfying the condition are constraints of the form $\mathsf{having}(Z)$. Each compound constraint $C_1 \Rightarrow C_2$ is thus, in fact, of the form $\mathsf{having}(Z) \Rightarrow C$. On one hand, this is restrictive. On the other hand, we show below that compound conditional constraints together with conjunctive constraints are rather general in terms of the ability to represent other constraints.

We now discuss the expressive power of constraints. In Section 7, we mentioned that constraints $\mathsf{model\text{-}of}(T)$ given by a set T of attribute implications are the most general ones. This follows from the fact that for each closure operator $C : 2^Y \to 2^Y$ representing a constraint there exists T such that $\mathsf{model\text{-}of}(T)$ represents the same constraint as C. For instance, one can take $T = \{A \Rightarrow C(A) \,|\, A \subseteq Y\}$ or any set of attribute implications which is semantically equivalent to T, e.g., an equivalent and non-redundant subset of T [8, 9, 10].

In a similar way as in mathematical logic where one can consider various systems of logical connectives (so-called adequate sets of connectives) which are adequate to describe all n-ary Boolean functions [16] (and thus all other connectives), we can consider systems of constrains which are adequate to describe all other constraints. Therefore, we might say that the system of constraints

$$\mathcal{M} = \{\mathsf{model\text{-}of}(T) \,|\, T \text{ is a set of AIs in } Y\} \tag{19}$$

is *adequate* because any closure operator C in Y belongs to $\mathcal{M}$. Since $\mathcal{M}$ consists solely of instances of $\mathsf{model\text{-}of}(T)$ (for all possible T's), constraints by models of attribute implications are the most general ones.

There are other systems of constrains that are adequate in the previous sense. For instance, one can consider

$$\mathcal{N} = \{\mathsf{being\text{-}intent\text{-}of}(K) \,|\, K \subseteq X' \times Y,\ X' \text{ is arbitrary}\},$$

which is also adequate. Indeed, for each closure operator C in Y, one can take $K \subseteq X' \times Y$ where $X' = \mathrm{fix}(C)$ and

$$K = \{\langle B, y\rangle \in X' \times Y \mid B \in X' \text{ and } y \in B\}.$$

The rest is obvious because $\mathrm{fix}(C)$ equals $\mathrm{fix}(^{\downarrow_K\uparrow_K})$.

One can form an adequate system of constraints using constraint $\mathsf{having}(B)$ by means of combination via implications and conjunctions as follows:

Theorem 19. *Let Y be a set of attributes and let*

$$\mathcal{H} = \{\mathsf{having}(A) \Rightarrow \mathsf{having}(B) \mid A, B \subseteq Y\},$$
$$\mathcal{I} = \{C_1 \,\&\, \cdots \,\&\, C_k \mid C_1, \ldots, C_k \in \mathcal{H} \text{ and } k \geq 1\}.$$

Then $\mathcal{I}$ is an adequate system of constraints.

Proof. Note that $\mathsf{having}(A) \Rightarrow \mathsf{having}(B)$ is as well-defined constraint since $\mathrm{fix}(\mathsf{having}(A))$ is a filter. Furthermore, each constraint $\mathsf{having}(A) \Rightarrow \mathsf{having}(B)$ is equivalent to $\mathsf{model\text{-}of}(\{A \Rightarrow B\})$. Indeed, for each $A, B \subseteq Y$, a set $M \subseteq Y$ is a fixed point of $\mathsf{having}(A) \Rightarrow \mathsf{having}(B)$ iff $M \in \mathrm{fix}(\mathsf{having}(A))$ implies $M \in \mathrm{fix}(\mathsf{having}(B))$, see Theorem 17. Since $M \in \mathrm{fix}(\mathsf{having}(A))$ means $A \subseteq M$ and analogously for B, we get that M is a fixed point of $\mathsf{having}(A) \Rightarrow \mathsf{having}(B)$ iff $M \models A \Rightarrow B$, i.e., M is a fixed point of $\mathsf{model\text{-}of}(\{A \Rightarrow B\})$.

The proof is finished as follows: For an arbitrary closure operator C, there exists T such that $\mathrm{fix}(C)$ equals $\mathrm{fix}(\mathsf{model\text{-}of}(T))$ because $\mathcal{M}$ defined by (19) is adequate. Let $T = \{A_i \Rightarrow B_i \mid i = 1, \ldots, k\}$ and let C_i be $\mathsf{having}(A_i) \Rightarrow \mathsf{having}(B_i)$ for $i = 1, \ldots, k$. Due to the previous observation, $C_1 \,\&\, \cdots \,\&\, C_k$ has the same fixed points as $\mathsf{model\text{-}of}(T)$, and thus the same fixed points as C, proving that $\mathcal{I}$ is adequate. □

9. Conclusions

We proposed a method that allows a user to impose constraints in formal concept analysis. The rationale of the method is to let the user specify additional knowledge, his background knowledge, he may have about the input data and extract from the data only the patterns that are compatible with the additional knowledge. The method proposed is suitable for dealing with background knowledge that can be represented by a closure operator in that being compatible with a background knowledge means being a fixed point of a given closure operator. The method subsumes a broad class of constraints. We provided foundations for imposing constraints on fomal concepts and attribute implications, algorithms for computing a constrained concept lattice and a base of constrained attribute implications, several particular constraints, and illustrative examples.

References

[1] Armstrong W. W.: Dependency structures in data base relationships. In: *IFIP Congress*, Geneva, Switzerland, pp. 580–583, 1974.

[2] Belohlavek R., Sklenar V.: Formal concept analysis constrained by attribute-dependency formulas. In: B. Ganter and R. Godin (Eds.): ICFCA 2005, *Lect. Notes Comp. Sci.* **3403**, pp. 176–191, Springer-Verlag, Berlin/Heidelberg, 2005.

[3] Belohlavek R., Vychodil V.: Formal concept analysis with constraints by closure operators. In: *Proc. ICCS 2006*, *LNCS* **4068**(2006), 131–143.

[4] Belohlavek R., Vychodil V.: Formal concept analysis with background knowledge: attribute priorities. *IEEE Trans. Systems, Man, and Cybernetics, Part C* **39**(4)(2009), 399–409.

[5] Belohlavek R., Vychodil V.: Background knowledge in formal concept analysis: constraints via closure operators. In: *Proc. ACM SAC 2010*, 1113–1114.

[6] Belohlavek R., Vychodil V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comput. System Sci.* **76**(1)(2010), 3–20.

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

[8] Ganter B.: *Two basic algorithms in concept analysis.* (Technical Report FB4-Preprint No. 831). TH Darmstadt, 1984.

[9] Ganter B., Wille R.: *Formal Concept Analysis. Mathematical Foundations.* Springer, Berlin, 1999.

[10] Guigues J.-L., Duquenne V.: Familles minimales d'implications informatives resultant d'un tableau de données binaires. *Math. Sci. Humaines* **95**(1986), 5–18.

[11] Johnson D. S, Yannakakis M., Papadimitriou C. H.: On generating all maximal independent sets. *Information Processing Letters* **27**(3)(1988), 119–123.

[12] Kuznetsov S.: A fast algorithm for computing all intersections of objects in a finite semi-lattice. *Automatic Documentation and Mathematical Linguistics*, **27**(5)(1993), 11–21.

[13] Kuznetsov S., Obiedkov S.: Comparing performance of algorithms for generating concept lattices. *J. Exp. Theor. Artif. Int.*, **14**(2002), 189–216.

[14] Lindig C.: Fast concept analysis. *Working with Conceptual Structures––Contributions to ICCS 2000,* pp. 152–161, 2000. Aachen: Shaker Verlag.

[15] Maier D.: *The Theory of Relational Databases.* Computer Science Press, Rockville, 1983.

[16] Mendelson E.: *Introduction to Mathematical Logic.* Chapman & Hall, London, fourth edition, 1997.

[17] Miettinen P., Mielikäinen T., Gionis A., Das G., Mannila H.: The discrete basis problem. *PKDD*, pp. 335–346, 2006. Springer.

[18] Norris E. M.: An Algorithm for Computing the Maximal Rectangles in a Binary Relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, **23**(2)(1978), 243–250.

[19] Pasquier N., Bastide Y., Taouil R., Lakhal L.: Efficient Mining of Association Rules Using Closed Itemset Lattices. *Information Systems* **24**(1)(1999), 25–46.

[20] Stumme G., Taouil R., Bastide Y., Pasquier N., Lakhal L.: Computing iceberg concept lattices with TITANIC, *Data Knowl. Eng.* **42**(2)(2002), 189–222.

[21] Xie Z.: Φ-Generalized concept lattice models: from power concepts to formal concepts, and then to robust concepts. Proc. CLA 2006, Int. Conference on Concept Lattices and Their Applications, pp. 219–230.

[22] Zaki M. J.: Mining non-redundant association rules. *Data Mining and Knowledge Discovery* **9**(2004), 223–248.

[23] Zhang C., Zhang S.: *Association Rule Mining. Models and Algorithms.* Springer, Berlin, 2002.