Trees in Concept Lattices*

Radim Belohlavek[1,3], Bernard De Baets[2], Jan Outrata[3], and Vilem Vychodil[3]

[1] Dept. Systems Science and Industrial Engineering
T. J. Watson School of Engineering and Applied Science
Binghamton University–SUNY, PO Box 6000, Binghamton, NY 13902–6000, USA
rbelohla@binghamton.edu
[2] Dept. Appl. Math., Biometrics, and Process Control, Ghent University
Coupure links 653, B-9000 Gent, Belgium
bernard.debaets@ugent.be
[3] Dept. Computer Science, Palacky University, Olomouc
Tomkova 40, CZ-779 00 Olomouc, Czech Republic
{jan.outrata, vilem.vychodil}@upol.cz

Abstract. The paper presents theorems characterizing concept lattices which happen to be trees after removing the bottom element. Concept lattices are the clustering/classification systems provided as an output of formal concept analysis. In general, a concept lattice may contain overlapping clusters and need not be a tree. On the other hand, tree-like classification schemes are appealing and are produced by several classification methods as the output. This paper attempts to help establish a bridge between concept lattices and tree-based classification methods. We present results presenting conditions for input data which are sufficient and necessary for the output concept lattice to form a tree after one removes its bottom element. In addition, we present illustrative examples and several remarks on related efforts and future research topics.

1 Introduction

Data tables describing objects and their attributes represent perhaps the most common form of data. Among several methods for analysis of object-attribute data, formal concept analysis (FCA) is becoming increasingly popular, see [7,4]. The main aim of FCA is to extract interesting clusters (called formal concepts) from tabular data along with a partial order of these clusters (called conceptual hierarchy). Formal concepts correspond to maximal rectangles in a data table and are easily interpretable by users. FCA is basically being used two ways. First, as a direct method of data analysis in which case the hierarchically ordered collection of formal concepts extracted from data is presented to a user/expert for further analysis, see e.g. [4] for such examples of FCA applications. Second,

* Supported by Kontakt 1–2006–33 (Bilateral Scientific Cooperation, project "Algebraic, logical and computational aspects of fuzzy relational modelling paradigms"), by grant No. 1ET101370417 of GA AV ČR, by grant No. 201/05/0079 of the Czech Science Foundation, and by institutional support, research plan MSM 6198959214.

V. Torra, Y. Narukawa, and Y. Yoshida (Eds.): MDAI 2007, LNAI 4617, pp. 174–184, 2007.

as a data preprocessing method in which case the extracted clusters are used for further processing, see e.g. [13] for applications of FCA in association rules mining.

Unlike several other clustering and classification techniques [1,5], which yield clustering and classification trees, FCA yields diagrams of hierarchically ordered clusters which are richer than trees. Namely, the diagrams are lattices and are called called concept lattices. A practical difference is that concept lattices usually contain overlapping clusters. Another difference is that the clusters in FCA are based on sharing of attributes rather than distance. FCA can thus be thought of as a new method of clustering and classification which is substantially different from conventional methods. Although FCA has been justified by many real-world applications already, see e.g. [4], the following quote from John Hartigan, a leading expert in clustering and classification, is relevant [1,5]:

> "My second remark is about future focus. We pay too much attention to the details of the algorithms. ... It is more important to think about the purposes of clustering, about the types of clusters we wish to construct, ... These details are interesting, ..., but we have plenty of algorithms already. ... what kinds of families of classes should we be looking for? At present, we think of partitions, trees, sometimes overlapping clusters; these structures are a faint echo of the rich classifications available in everyday language. ... We must seek sufficiently rich class of structures ..."

The present paper seeks to contribute to the problem of establishing relationships between FCA and other methods of clustering and classification. Needless to say, this goal requires a long-term effort. In this paper we consider a particular problem. Namely, we present conditions for input data which are sufficient and necessary for the output concept lattice to form a tree after removing its bottom element. In addition, we present illustrative examples and several remarks on related efforts and future research topics. Note that a related problem, namely, of selecting a tree from a concept lattice by means of constraints using attribute-dependency formulas, was considered in [2].

Section 2 presents preliminaries. Section 3 presents the main results, illustrative examples, and remarks. Section 4 presents conclusions and an outline of future research.

2 Preliminaries

In this section, we summarize basic notions of formal concept analysis (FCA). An object-attribute data table describing which objects have which attributes can be identified with a triplet $\langle X, Y, I \rangle$ where X is a non-empty set (of objects), Y is a non-empty set (of attributes), and $I \subseteq X \times Y$ is an (object-attribute) relation. In FCA, $\langle X, Y, I \rangle$ is called a formal context. Objects and attributes correspond to table rows and columns, respectively, and $\langle x, y \rangle \in I$ indicates that object x has attribute y (table entry corresponding to row x and column y contains $\times$ or

1; if $\langle x, y\rangle \notin I$ the table entry contains blank symbol or 0). For each $A \subseteq X$ and $B \subseteq Y$ denote by $A^{\uparrow}$ a subset of Y and by $B^{\downarrow}$ a subset of X defined by

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

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

Described verbally, $A^{\uparrow}$ is the set of all attributes from Y shared by all objects from A and $B^{\downarrow}$ is the set of all objects from X sharing all attributes from B. A formal concept in $\langle X, Y, I\rangle$ is a pair $\langle A, B\rangle$ of $A \subseteq X$ and $B \subseteq Y$ satisfying $A^{\uparrow} = B$ and $B^{\downarrow} = A$. That is, a formal concept 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 (submatrices) 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 $A \times B' \subseteq I$.

A set $\mathcal{B}(X, Y, I) = \{\langle A, B\rangle \mid A^{\uparrow} = B, B^{\downarrow} = A\}$ of all formal concepts in data $\langle X, Y, I\rangle$ can be equipped with a partial order $\leq$ (modeling the subconcept-superconcept hierarchy, e.g. $\mathsf{dog} \leq \mathsf{mammal}$) defined by

$$\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle \quad \text{iff} \quad A_1 \subseteq A_2 \ (\text{iff } B_2 \subseteq B_1). \tag{3}$$

Under $\leq$, $\mathcal{B}(X, Y, I)$ happens to be a complete lattice, called a concept lattice of $\langle X, Y, I\rangle$, the basic structure of which is described by a so-called main theorem of concept lattices [7]:

Theorem 1 (Main Theorem of Concept Lattices). (1) *The set $\mathcal{B}(X, Y, I)$ is under $\leq$ a complete lattice where the infima and suprema are given by*

$$\textstyle\bigwedge_{j\in J}\langle A_j, B_j\rangle = \big\langle \bigcap_{j\in J} A_j, (\bigcup_{j\in J} B_j)^{\downarrow\uparrow}\big\rangle, \tag{4}$$

$$\textstyle\bigvee_{j\in J}\langle A_j, B_j\rangle = \big\langle (\bigcup_{j\in J} A_j)^{\uparrow\downarrow}, \bigcap_{j\in J} B_j\big\rangle. \tag{5}$$

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

(i) *$\gamma(X)$ is $\bigvee$-dense in V, $\mu(Y)$ is $\bigwedge$-dense in V;*

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

Note that a subset $K \subseteq V$ is $\bigvee$-dense in V ($\bigwedge$-dense in V) if for every $v \in V$ there is $K' \subseteq K$ such that $v = \bigvee K'$ ($v = \bigwedge K'$). Note also that operators $^{\uparrow}$ and $^{\downarrow}$ form a so-called Galois connection [7] and that $\mathcal{B}(X, Y, I)$ is in fact a set of all fixed points of $^{\uparrow}$ and $^{\downarrow}$. That is, $^{\uparrow}$ and $^{\downarrow}$ satisfy the following conditions:

$$A \subseteq A^{\uparrow\downarrow}, \tag{6}$$

$$\text{if } A_1 \subseteq A_2 \text{ then } A_2{}^{\uparrow} \subseteq A_1{}^{\uparrow}, \tag{7}$$

$$B \subseteq B^{\downarrow\uparrow}, \tag{8}$$

$$\text{if } B_1 \subseteq B_2 \text{ then } B_2{}^{\downarrow} \subseteq B_1{}^{\downarrow}, \tag{9}$$

for each $A, A_1, A_2 \subseteq X$ and $B, B_1, B_2 \subseteq Y$. Furthermore, the composed operators $^{\uparrow\downarrow} : 2^X \to 2^X$ and $^{\downarrow\uparrow} : 2^Y \to 2^Y$ are closure operators in X and Y,

respectively. As a consequence, $A \subseteq X$ is an extent of some concept in $\mathcal{B}(X, Y, I)$ (i.e., there is $B \subseteq Y$ such that $\langle A, B\rangle \in \mathcal{B}(X, Y, I)$) iff $A = A^{\uparrow\downarrow}$, i.e. A is closed under $^{\uparrow\downarrow}$. Analogously for intents.

Concept lattices are the primary output of formal concept analysis. There is another output of FCA which is equally important, namely, so-called non-redundant bases of attribute implications. An attribute implication is an expression $A \Rightarrow B$ where $A, B \subseteq Y$ with Y being the same set of attributes as above. An attribute implication $A \Rightarrow B$ is called true in $M \subseteq Y$, written $M \models A \Rightarrow B$, if the following condition is satisfied:

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

If $M \subseteq Y$ represents a set of attributes of some object x then $M \models A \Rightarrow B$ has the following meaning: "if x has all attributes from A, then x has all attributes from B". Thus, attribute implications are particular if-then rules describing dependencies between attributes.

Given a formal context $\langle X, Y, I\rangle$, for each $x \in X$ we define a set I_x of attributes $I_x = \{y \in Y \mid \langle x, y\rangle \in I\}$, i.e. I_x is the set of all attributes of object x in $\langle X, Y, I\rangle$. Notice that I_x corresponds to a row in data table representing format context $\langle X, Y, I\rangle$. An attribute implication $A \Rightarrow B$ is called true in $\langle X, Y, I\rangle$, written $I \models A \Rightarrow B$, iff $I_x \models A \Rightarrow B$ for each $(x \in X)$. Hence, $I \models A \Rightarrow B$ iff for each object $x \in X$ we have: if x has all attributes from A, then x has all attributes from B.

The set of all attribute implications which are true in $\langle X, Y, I\rangle$ is, along with the concept lattice $\mathcal{B}(X, Y, I)$, the basic output of FCA. Unfortunately, the set of all attribute implications is usually too large and it cannot be presented to users directly. Therefore, we use special indirect description of all attribute implications being true in $\langle X, Y, I\rangle$. Namely, we select from all the attribute implications in question a small subset from which the other implications follow. This can be done using the following notions.

Let T be any set of attribute implications. A set $M \subseteq Y$ of attributes is called a model of T, if $M \models A \Rightarrow B$ for each $A \Rightarrow B \in T$. The set of all models of T will be denoted by $\mathrm{Mod}(T)$, i.e.

$$\mathrm{Mod}(T) = \{M \subseteq Y \mid \text{for each } A \Rightarrow B \in T\colon M \models A \Rightarrow B\}. \tag{10}$$

An attribute implication $A \Rightarrow B$ follows from T ($A \Rightarrow B$ is semantically entailed by T), written $T \models A \Rightarrow B$, if $M \models A \Rightarrow B$ for each $M \in \mathrm{Mod}(T)$. A set T of attribute implications is called complete in $\langle X, Y, I\rangle$ if, for each attribute implication $A \Rightarrow B$, we have

$$T \models A \Rightarrow B \quad \text{iff} \quad I \models A \Rightarrow B,$$

i.e., if the attribute implications which are entailed by T are exactly the attribute implications which are true in $\langle X, Y, I\rangle$. Hence, if T is complete in $\langle X, Y, I\rangle$, then T describes exactly the attribute implications which are true in $\langle X, Y, I\rangle$. This is important especially if T is "reasonably small". Therefore, we define the following notion. A set T of attribute implications is a non-redundant basis of $\langle X, Y, I\rangle$ if (i) T is complete in $\langle X, Y, I\rangle$ and (ii) no proper subset of T is

complete in $\langle X, Y, I\rangle$. Alternatively, a non-redundant basis of $\langle X, Y, I\rangle$ can be described as complete sets of attribute implications such that no implication in the set is entailed by the other implications in that set. There have been proposed algorithms to generate, given $\langle X, Y, I\rangle$, a non-redundant basis of $\langle X, Y, I\rangle$, see e.g. [6,7,10].

For a detailed information on formal concept analysis and lattice theory we refer to [4,7,8] where a reader can find theoretical foundations, methods and algorithms, and applications in various areas.

3 Trees in Concept Lattices

In this section we will be interested in concept lattices corresponding to trees. Trees are usually defined as undirected graphs which are acyclic and connected [9]. Since we are going to identify trees in particular ordered sets, we deal with trees as with ordered sets. In particular, a finite partially ordered set $\langle U, \leq\rangle$ will be called a tree if for each $a, b \in U$:

(i) there is a supremum of a and b in $\langle U, \leq\rangle$, and
(ii) there is an infimum of a and b in $\langle U, \leq\rangle$ iff a and b are comparable (i.e., iff $a \leq b$ or $b \leq a$).

Obviously, $\langle U, \leq\rangle$ being a tree corresponds to the usual graph-theoretical representation of a rooted tree. The root of $\langle U, \leq\rangle$ is the supremum of all elements from U (which exists in $\langle U, \leq\rangle$ because U is finite). An element $u \in U$ is a direct descendant of $w \in U$ iff $u < w$, and there is no $v \in U$ such that $u < v < w$.

From Theorem 1 it follows that each concept lattice is a complete lattice. Hence, the above-mentioned condition (i) is satisfied for each $\mathcal{B}(X, Y, I)$. On the other hand, (ii) need not be satisfied. It is easily seen that (ii) is satisfied iff $\mathcal{B}(X, Y, I)$ is linearly ordered. So, the whole concept lattice is a tree iff it is linear, which is not a worthwhile observation because linear trees are a degenerate form of trees and therefore not interesting. Because of the observation we have just made, we turn our attention to trees which form important parts of concept lattices. We focus mainly on trees which appear in $\mathcal{B}(X, Y, I)$ if we remove its least element.

Since concept lattices are complete lattices, each concept lattice $\mathcal{B}(X, Y, I)$ has both the greatest and least element. Namely, $\langle X, X^{\uparrow}\rangle$ is the greatest element (concept of all objects) of $\mathcal{B}(X, Y, I)$ and $\langle Y^{\downarrow}, Y\rangle$ is the least one (concept of objects sharing all attributes from Y). If $\langle X, Y, I\rangle$ does not contain an attribute shared by all objects (i.e., a table representing $\langle X, Y, I\rangle$ does not contain a column full of $\times$'s), which is quite common if $\langle X, Y, I\rangle$ represents a real-world data, then $\langle X, X^{\uparrow}\rangle$ equals $\langle X, \emptyset\rangle$. Analogously, there is no object sharing all the attributes from Y (i.e., a table representing $\langle X, Y, I\rangle$ does not contain a row full of $\times$'s), $\langle Y^{\downarrow}, Y\rangle$ becomes $\langle \emptyset, Y\rangle$.

In what follows we investigate under which conditions $\mathcal{B}(X, Y, I)$ becomes a tree if we remove its least element.

3.1 Formal Contexts Generating Trees

For brevity, let $\mathcal{B}(X,Y,I)-\{\langle Y^{\downarrow},Y\rangle\}$ be denoted by $\mathcal{B}^{\curlywedge}(X,Y,I)$. Note that if we consider $\mathcal{B}^{\curlywedge}(X,Y,I)$, we assume that it is equipped with a partial order which is a restriction of the partial order defined by (3) to elements of $\mathcal{B}^{\curlywedge}(X,Y,I)$.

The following assertion characterizes when $\mathcal{B}^{\curlywedge}(X,Y,I)$ is a tree in terms of extents of formal concepts.

Theorem 2. *Let $\langle X,Y,I\rangle$ be a formal context. Then $\mathcal{B}^{\curlywedge}(X,Y,I)$ is a tree iff, for each concepts $\langle A,B\rangle,\langle C,D\rangle\in\mathcal{B}(X,Y,I)$ at least one of the following is true:*

(i) $A\subseteq C$ *or* $C\subseteq A$,
(ii) $A\cap C\subseteq Y^{\downarrow}$.

Proof. Let $\mathcal{B}^{\curlywedge}(X,Y,I)$ be a tree. Take any concepts $\langle A,B\rangle,\langle C,D\rangle\in\mathcal{B}(X,Y,I)$. If (i) is satisfied for A and C, we are done. Hence, assume that (i) is not satisfied, i.e. we have $A\nsubseteq C$ and $C\nsubseteq A$. From the definition of $\leq$, see (3), it follows that $\langle A,B\rangle\not\leq\langle C,D\rangle$ and $\langle C,D\rangle\not\leq\langle A,B\rangle$, i.e. formal concepts $\langle A,B\rangle$ and $\langle C,D\rangle$ are incomparable. Therefore, both $\langle A,B\rangle$ and $\langle C,D\rangle$ are in $\mathcal{B}^{\curlywedge}(X,Y,I)$. Since $\mathcal{B}^{\curlywedge}(X,Y,I)$ is supposed to be a tree, infimum of $\langle A,B\rangle$ and $\langle C,D\rangle$ does not exist in $\mathcal{B}^{\curlywedge}(X,Y,I)$. It means that the infimum of $\langle A,B\rangle$ and $\langle C,D\rangle$ in $\mathcal{B}(X,Y,I)$ is $\langle Y^{\downarrow},Y\rangle$ because $\mathcal{B}^{\curlywedge}(X,Y,I)$ results from $\mathcal{B}(X,Y,I)$ by removing $\langle Y^{\downarrow},Y\rangle$ and $\mathcal{B}(X,Y,I)$ is a complete lattice. Using (4), we get that $Y^{\downarrow}=A\cap C$, showing (ii).

Conversely, let (i) and (ii) be satisfied for any $\langle A,B\rangle,\langle C,D\rangle\in\mathcal{B}(X,Y,I)$. Take $\langle A,B\rangle,\langle C,D\rangle\in\mathcal{B}^{\curlywedge}(X,Y,I)$ such that $\langle A,B\rangle$ and $\langle C,D\rangle$ are incomparable. Such $\langle A,B\rangle$ and $\langle C,D\rangle$ cannot satisfy (i), i.e. we have $A\cap C\subseteq Y^{\downarrow}$. Hence, using (4), the infimum of $\langle A,B\rangle$ and $\langle C,D\rangle$ in $\mathcal{B}(X,Y,I)$ is the least element of $\mathcal{B}(X,Y,I)$. As a consequence, $\langle A,B\rangle$ and $\langle C,D\rangle$ does not have an infimum in $\mathcal{B}^{\curlywedge}(X,Y,I)$ which proves that $\mathcal{B}^{\curlywedge}(X,Y,I)$ is a tree. □

Theorem 2 can also be formulated in terms of intents of formal concepts:

Corollary 1. *$\mathcal{B}^{\curlywedge}(X,Y,I)$ is a tree iff, for each $\langle A,B\rangle,\langle C,D\rangle\in\mathcal{B}(X,Y,I)$ we either have* (i) $B\subseteq D$ *or* $D\subseteq B$, *or* (ii) $(B\cup D)^{\downarrow}\subseteq Y^{\downarrow}$. □

Remark 1. If $\langle Y^{\downarrow},Y\rangle$ is equal to $\langle\emptyset,Y\rangle$, i.e. if the table representing $\langle Y^{\downarrow},Y\rangle$ does not contain a row full of $\times$'s (or 1's), then (iii) in Theorem 2 simplifies to $A\cap C=\emptyset$, i.e. A and C are required to be disjoint.

Example 1. Consider a set of objects $X=\{1,2,\ldots,14\}$ (objects are denoted by numbers) and a set of attributes $Y=\{g,h,\ldots,z\}$. If we consider a formal context $\langle X,Y,I\rangle$ which is represented by the data table in Fig. 1 (left) then the corresponding $\mathcal{B}^{\curlywedge}(X,Y,I)$, which is depicted in Fig. 1 (right), is a tree. The root of the tree represents concept $\langle X,\emptyset\rangle$. The other nodes are numbered and the intents of the corresponding concepts are the following:

1: $\{i,r\}$,	5: $\{m,s,z\}$,	9: $\{g,m,n,q,s,v,z\}$,
2: $\{i,o,r\}$,	6: $\{g,m,n,s,v,z\}$,	10: $\{m,s,x,z\}$,
3: $\{i,l,o,r\}$,	7: $\{g,j,m,n,p,s,t,v,z\}$,	11: $\{h,m,s,x,z\}$,
4: $\{i,r,w\}$,	8: $\{g,j,k,m,n,p,s,t,u,v,z\}$,	12: $\{m,s,y,z\}$.

	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1.	1	0	0	0	0	0	1	1	0	0	1	0	1	0	0	1	0	0	0	1
2.	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1
3.	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
4.	1	0	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0	0	1
5.	1	0	0	1	1	0	1	1	0	1	0	0	1	1	1	1	0	0	0	1
6.	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	1	0	0	0	1
7.	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	1
8.	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
9.	1	0	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0	0	1
10.	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
11.	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	1
12.	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
13.	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	1
14.	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

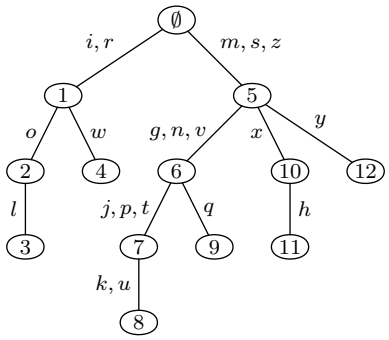


Fig. 1. Formal context generating a tree

If two nodes are connected by an edge, the lower concept has a strictly greater intent. Using this observation, we can decorate edges of the tree by attributes being added to intents of lower concepts as it is shown in Fig. 1 (right). On can check that all the intents from $\mathcal{B}(X, Y, I)$ satisfy condition of Theorem 1.

Now, an important question is, if we can check that $\mathcal{B}^{\curlywedge}(X, Y, I)$ is a tree directly from the context $\langle X, Y, I\rangle$, i.e. without computing the set of all concepts first. We shall show that this is indeed possible. We will take advantage of the following notion.

Definition 1. *Let $\langle X, Y, I\rangle$ be a formal context. We say that $\langle X, Y, I\rangle$ generates a tree if $\mathcal{B}^{\curlywedge}(X, Y, I)$ is a tree.*

Recall that due to (2), $\{y\}^{\downarrow}$ is a set of all objects sharing the attribute y. That is, $\{y\}^{\downarrow}$ naturally corresponds to a column in data table representing $\langle X, Y, I\rangle$. Such "columns" will play an important role in the following theorem which characterizes contexts generating trees.

Theorem 3. *Let $\langle X, Y, I\rangle$ be a formal context. Then $\langle X, Y, I\rangle$ generates a tree iff, for any attributes $y_1, y_2 \in Y$, at least one of the following conditions is true:*

(i) $\{y_1\}^{\downarrow} \subseteq \{y_2\}^{\downarrow}$,
(ii) $\{y_2\}^{\downarrow} \subseteq \{y_1\}^{\downarrow}$,
(iii) $\{y_1\}^{\downarrow} \cap \{y_2\}^{\downarrow} \subseteq Y^{\downarrow}$.

Proof. Assume that $\langle X, Y, I\rangle$ generates a tree, i.e. $\mathcal{B}^{\curlywedge}(X, Y, I)$ is a tree. Each pair of the form $\langle \{y\}^{\downarrow}, \{y\}^{\downarrow\uparrow}\rangle$ is a formal concept from $\mathcal{B}(X, Y, I)$, see [7]. Therefore, Theorem 2 yields that the above conditions (i)–(iii), being particular instances of (i) and (ii) from Theorem 2, are satisfied.

Conversely, suppose that $\langle X, Y, I\rangle$ does not generate a tree. Thus, there are some incomparable formal concepts $\langle A_1, B_1\rangle, \langle A_2, B_2\rangle \in \mathcal{B}(X, Y, I)$ whose infimum is not equal to the least element of $\mathcal{B}(X, Y, I)$. That is, for $\langle A_1, B_1\rangle$ and

$\langle A_2, B_2\rangle$ we have $A_1 \not\subseteq A_2$, $A_2 \not\subseteq A_1$, and $A_1 \cap A_2 \not\subseteq Y^{\downarrow}$. Note that as a consequence we get that $B_1 \not\subseteq B_2$ and $B_2 \not\subseteq B_1$. We now show that we can pick from B_1 and B_2 two attributes violating the above conditions (i)–(iii). Since $B_1 \not\subseteq B_2$, there is $y_1 \in B_1$ such that $y_1 \notin B_2$. Analogously, there is $y_2 \in B_2$ such that $y_2 \notin B_1$ because $B_2 \not\subseteq B_1$. For y_1 and y_2 we can show that (i) is not satisfied. Indeed, from $y_1 \in B_1 = {A_1}^{\uparrow}$ and (9) it follows that

$$A_1 \subseteq \{y_1\}^{\downarrow}. \tag{11}$$

Moreover, $y_2 \notin B_1 = {A_1}^{\uparrow}$ gives that there is $x \in A_1$ such that $\langle x, y_2\rangle \notin I$. Hence, there is $x \in A_1$ such that $x \notin \{y_2\}^{\downarrow}$, i.e. we get

$$A_1 \not\subseteq \{y_2\}^{\downarrow}. \tag{12}$$

As an immediate consequence of (11) and (12) we get that $\{y_1\}^{\downarrow} \not\subseteq \{y_2\}^{\downarrow}$, i.e. condition (i) is violated. In a symmetric way (i.e., with y_1 and y_2 interchanged), we can also show that (ii) is violated. So, now it remains to show that (iii) cannot be satisfied. But this is now easy to see. From $y_1 \in B_1$, $y_2 \in B_2$, and (9) we get $A_1 = {B_1}^{\downarrow} \subseteq \{y_1\}^{\downarrow}$ and $A_2 = {B_2}^{\downarrow} \subseteq \{y_2\}^{\downarrow}$ which yield $A_1 \cap A_2 \subseteq \{y_1\}^{\downarrow} \cap \{y_2\}^{\downarrow}$. Therefore, from $A_1 \cap A_2 \not\subseteq Y^{\downarrow}$ it follows that $\{y_1\}^{\downarrow} \cap \{y_2\}^{\downarrow} \not\subseteq Y^{\downarrow}$, showing that (iii) is not satisfied. Altogether, we have shown that if $\langle X, Y, I\rangle$ does not generate a tree, then there are $y_1, y_2 \in Y$ such that neither of (i)–(iii) is satisfied. □

Remark 2. Conditions (i)–(iii) from Theorem 3 say that, roughly speaking, for each two columns of a data table, either one of the columns is contained in the other, or the columns have in common only attributes shared by all objects. In particular, if no row of the data table contains all ×'s (or 1's), the latter condition says that the columns do not have any attributes in common. Note that (i)–(iii) can be checked with asymptotic time complexity $O(n^3)$, where n is the maximum of $|X|$ and $|Y|$.

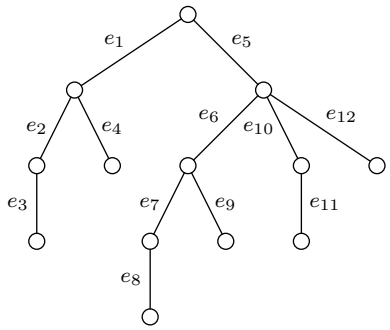
Theorem 3 can be restated as follows:

Corollary 2. *A formal context $\langle X, Y, I\rangle$ generates a tree iff, for any $y_1, y_2 \in Y$, we either have $\{y_1\}^{\downarrow} \cap \{y_2\}^{\downarrow} \in \{\{y_1\}^{\downarrow}, \{y_2\}^{\downarrow}\}$, or $\{y_1\}^{\downarrow} \cap \{y_2\}^{\downarrow} \subseteq Y^{\downarrow}$.* □

We now turn our attention to a converse problem. Given a tree (defined possibly by its graph-theoretical representation), we wish to find a formal context which generates the tree. First, let us note that for each tree such a context exists. This is, in fact, a consequence of the main theorem of concept lattices. In a more detail, consider a graph $\mathbf{G} = \langle V, E\rangle$ which is a tree [9]. We say that edge $e_1 \in E$ in under $e_2 \in E$ (in $\mathbf{G}$) if $\mathbf{G}$ contains a path $v_1, e_1, \ldots, v_2, e_2, \ldots$ ending in the root node of $\mathbf{G}$ (for the notions involved, see [9]). We now get the following characterization.

Theorem 4. *Let $\mathbf{G} = \langle V, E\rangle$ be a tree. Define a formal context $\langle E, E, I_{\mathbf{G}}\rangle$ such that $\langle e_1, e_2\rangle \in I_{\mathbf{G}}$ iff e_1 is under e_2 in $\mathbf{G}$. Then $\langle E, E, I_{\mathbf{G}}\rangle$ generates a tree which is isomorphic to $\mathbf{G} = \langle V, E\rangle$.*

Proof. Follows from Theorem 1. We omit details of the proof due to the limited scope of this paper. □



	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}
e_1	1	0	0	0	0	0	0	0	0	0	0	0
e_2	1	1	0	0	0	0	0	0	0	0	0	0
e_3	1	1	1	0	0	0	0	0	0	0	0	0
e_4	1	0	0	1	0	0	0	0	0	0	0	0
e_5	0	0	0	0	1	0	0	0	0	0	0	0
e_6	0	0	0	0	1	1	0	0	0	0	0	0
e_7	0	0	0	0	1	1	1	0	0	0	0	0
e_8	0	0	0	0	1	1	1	1	0	0	0	0
e_9	0	0	0	0	1	1	0	0	1	0	0	0
e_{10}	0	0	0	0	1	0	0	0	0	1	0	0
e_{11}	0	0	0	0	1	0	0	0	0	1	1	0
e_{12}	0	0	0	0	1	0	0	0	0	0	0	1

Fig. 2. Tree and its generating formal context

Example 2. If we return to Example 1 and consider the tree from Fig. 1 (right) as an input tree, we may construct a formal context generating that tree as follows. First, we choose a labeling of edges. For instance, we may choose labeling as in Fig. 2 (left). Then, a formal context which corresponds to $\langle E, E, I_{\mathbf{G}} \rangle$ from Theorem 4 is given by data table in Fig. 2 (right). Since we have labeled the edges in a depth-first manner, $\langle E, E, I_{\mathbf{G}} \rangle$ is in a lower-triangular form. By Theorem 4, tree $\mathcal{B}^{\lambda}(E, E, I_{\mathbf{G}})$ generated from $\langle E, E, I_{\mathbf{G}} \rangle$ is isomorphic to the initial tree.

3.2 Characterization of Trees by Attribute Implications

In the previous section, we have shown that contexts generating trees can be characterized based on the dependencies between attributes (columns of data tables representing formal contexts). Since attribute dependencies are often expressed by attribute implications, it is tempting to look at trees in a concept lattice from the point of view of attribute implications.

The following assertion characterizes contexts generating trees by means of attribute implications.

Theorem 5. *Let $\langle X, Y, I \rangle$ be a formal context. Then $\langle X, Y, I \rangle$ generates a tree iff, for any attributes $y_1, y_2 \in Y$, at least one of the following is true:*

(i) $I \models \{y_1\} \Rightarrow \{y_2\}$,
(ii) $I \models \{y_2\} \Rightarrow \{y_1\}$,
(iii) $I \models \{y_1, y_2\} \Rightarrow Y$.

Proof. Note that attribute implications being true in $\langle X, Y, I \rangle$ can be characterized using operators $^{\uparrow}$ and $^{\downarrow}$ induced by $\langle X, Y, I \rangle$. Namely, one can check that $I \models A \Rightarrow B$ iff, for each $x \in X$, if $A \subseteq \{x\}^{\uparrow}$ then $B \subseteq \{x\}^{\uparrow}$ which is iff, for each $x \in X$, if $x \in A^{\downarrow}$ then $x \in B^{\downarrow}$ which is true iff $A^{\downarrow} \subseteq B^{\downarrow}$, see [7]. Thus, (i) and (ii) are true iff $\{y_1\}^{\downarrow} \subseteq \{y_2\}^{\downarrow}$ and $\{y_2\}^{\downarrow} \subseteq \{y_1\}^{\downarrow}$, cf. Theorem 3 (i) and (ii). Moreover, (iii) is true iff we have $\{y_1\}^{\downarrow} \cap \{y_2\}^{\downarrow} = (\{y_1\} \cup \{y_2\})^{\downarrow} = \{y_1, y_2\}^{\downarrow} \subseteq Y^{\downarrow}$. Using Theorem 3, we finally obtain that $\langle X, Y, I \rangle$ generates a tree iff, for any $y_1, y_2 \in Y$, at least one of (i)–(iii) is true. □

3.3 Algorithms For Trees in Concept Lattices

Trees in concept lattices, as they were introduced in previous sections, can be computed by algorithms for computing formal concepts. Currently, there have been proposed several algorithms, see e.g. [4,7,12] and a survey paper [11]. Some of the algorithms for FCA simplify in case of contexts generating trees.

For instance, Lindig's algorithm for generating formal concepts simplifies due to the fact that it is no longer necessary to organize found concepts in some type of searching structure, because we cannot generate the same concept multiple times. Indeed, recall from [12] that Lindig's algorithm is based on the NextNeighbors procedure which, given a concept as its input, generates all its (lower or upper) neighbors. Then, all concepts are computed using a recursive procedure which fist uses NextNeighbors to compute neighbors of a given concept and then recursively processes all the neighbors to obtain further concepts. During the computation, the original procedure has to ensure that no concept will be computed twice (or multiple times). Therefore, the procedure must organize all found concepts in a suitable searching structure which allows us to check whether a concept has already been found. Needless to say, the searching structure should be efficient because the tests of presence of a concept between the found concepts influences the overall efficiency of the procedure. The searching structure is usually implemented as a searching tree or a hashing table.

In case of context generating trees, this part of the algorithm need not be implemented at all because the only concept that can be computed multiple times is $\langle Y^{\downarrow}, Y \rangle$ which is excluded from $\mathcal{B}^{\wedge}(X, Y, I)$. This allows to have an algorithm which is faster and simpler to implement.

4 Conclusions and Future Research

We presented conditions for input data for FCA which are sufficient and necessary for the output concept lattice to form a tree after one removes its bottom element. Trees are the most common structures which appear in traditional clustering and classification. Out long-term effort will be focused on establishing connections between FCA and other clustering and classification methods. First, establishing such relationships helps us see the pros and cons, and limits of the respective methods. Second, with the basic relationships established, one can hopefully enrich the respective methods by techniques used in the other methods. The problems we want to address next include the following ones:

- A concept lattice can be seen as consisting of several overlapping trees. What can we say about such a "decomposition" of a concept lattice into trees? What are the relationships between these trees?
- A user of FCA might be interested in a part of a concept lattice rather than in the whole lattice. Particularly, that part might be a tree, but other parts might be interesting as well. The issue of selecting parts of concept lattices by constraints was discussed in [2,3]. In particular, it can be shown that a tree contained in a concept lattice can be selected by means of a particular

closure operator. Constraints which lead to tree-like parts of concept lattices need to be further investigated.
– Investigate connections between concept lattices and decision trees. Both concept lattices and decision trees contain clusters of objects in their nodes. Leafs of a decision tree correspond to particular attribute-concepts. A construction of a decision tree may be thought of as a selection of a particular part from a concept lattice. Containment of decision trees in concept lattices needs to be further investigated.

References

1. Arabie, P., Hubert, L.J., DeSoete, G.: Clustering and Classification. World Scientific (1996)
2. Belohlavek, R., Sklenar, V.: Formal concept analysis constrained by attribute-dependency formulas. In: Ganter, B., Godin, R. (eds.) ICFCA 2005. LNCS (LNAI), vol. 3403, pp. 176–191. Springer, Heidelberg (2005)
3. Belohlavek, R., Vychodil, V.: Formal concept analysis with constraints by closure operators. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 131–143. Springer, Heidelberg (2006)
4. Carpineto, C., Romano, G.: Concept Data Analysis. Theory and Applications. J. Wiley, Chichester (2004)
5. Everitt, B.S., Landau, S., Leese, M.: Cluster Analysis, 4th edn. Oxford University Press, New York (2001)
6. Ganter, B.: Begriffe und Implikationen, manuscript (1998)
7. Ganter, B., Wille, R.: Formal Concept Analysis. Mathematical Foundations. Springer, Heidelberg (1999)
8. Gratzer, G.A.: General Lattice Theory. 2nd edn. Birkhauser (1998)
9. Grimaldi, R.P.: Discrete and Combinatorial Mathematics, Pearson Education, 5th edn. (2002)
10. Guigues, J.-L., Duquenne, V.: Familles minimales d'implications informatives resultant d'un tableau de données binaires. Math. Sci. Humaines 95, 5–18 (1986)
11. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. J. Exp. Theor. Artif. Intelligence 14(2/3), 189–216 (2002)
12. Lindig, C.: Fast concept analysis. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS, vol. 1867, pp. 152–161. Springer, Heidelberg (2000)
13. Zaki, M.J.: Mining non-redundant association rules. Data Mining and Knowledge Discovery 9, 223–248 (2004)