Lindig's Algorithm for Concept Lattices over Graded Attributes*

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. Formal concept analysis (FCA) is a method of exploratory data analysis. The data is in the form of a table describing relationship between objects (rows) and attributes (columns), where table entries are grades representing degrees to which objects have attributes. The main output of FCA is a hierarchical structure (so-called concept lattice) of conceptual clusters (so-called formal concepts) present in the data. This paper focuses on algorithmic aspects of FCA of data with graded attributes. Namely, we focus on the problem of generating efficiently all clusters present in the data together with their subconcept-superconcept hierarchy. We present theoretical foundations, the algorithm, analysis of its efficiency, and comparison with other algorithms.

1 Introduction

Our paper contributes to the area of exploratory analysis of tabular data. Namely, we focus on data supplied as tables with rows corresponding to objects and columns corresponding to attributes. Datasets of this form are often described by bivalent (presence/absence) attributes. That is, each attribute either applies or does not apply to a particular object. The corresponding data table is thus binary, i.e., it is a matrix filled with 0's (the object given by row does not have the attribute given by column) and 1's (the object given by row has the attribute given by column). Formal concept analysis (FCA) [8,20] aims at revealing all "conceptual clusters" (so-called formal concepts) which are hidden in the binary data table. Formal concepts are particular clusters of objects and

* 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. 156–167, 2007.

attributes that correspond to maximal submatrices filled with 1's. Alternatively, formal concepts may be understood as concepts in the traditional sense—as entities consisting of a set A of objects and a set B of attributes to which the concept applies (e.g., concept dog applies to objects poodle, foxhound, ... and attributes barks, has limbs, ...). Formal concepts are partially ordered by the subconcept-superconcept hierarchy. The resulting partially ordered set of formal concepts (the so-called concept lattice) represents a hierarchical structure of all naturally interpretable clusters (concepts) existing in the data (e.g., concept mammal is more general a concept than dog). Applications of FCA in exploratory data analysis can be found in [6,8]; [8] provides theoretical foundations.

In practice, more often than not, attributes are graded (fuzzy) rather than bivalent. That is, an attribute applies to an object to a certain degree which may be represented, e.g., by a number from the unit interval $[0, 1]$. The data table is then a $[0, 1]$-valued matrix, with table entries corresponding to degrees to which attributes apply to objects. There have been several approaches to FCA in a graded setting. The most relevant approach was independently developed in [1,2,4] and [19]. Up to now, not much attention has been paid to computational aspects of FCA with graded attributes. As an exception, in [3] the author presents an algorithm for determining formal concepts present in data table with graded attributes.

The aim of this paper is to propose another algorithm for generating all formal concepts (which can be seen again as certain maximal submatrices). Unlike the algorithm presented in [3], our algorithm enables us to generate all formal concepts together with their subconcept-superconcept hierarchy. The absence of conceptual hierarchy is not crucial if, for instance, the output of FCA is used for preprocessing (e.g., for mining non-redundant association rules, see [21]). On the other hand, if we want to present the output of analysis directly to users, it is more convenient to depict the clusters in a hierarchy which models the natural subconcept-superconcept ordering. Thus, from the point of view of applications, it is important to have an efficient algorithm which generates the hierarchy along with the clusters present in data table with graded attributes.

The paper is organized as follows. In Section 2, we present preliminaries from classical formal concept analysis, fuzzy sets, and formal concept analysis of data with graded attributes. In Section 3, we present the algorithm and prove its correctness. Section 4 contains experiments and comparisons of the new algorithm with that from [3].

2 Preliminaries

This section provides basic notions of FCA and fuzzy logic. More details can be found in [6,8,20] (formal concept analysis) and in [1,9,12,16] (fuzzy logic).

2.1 Formal Concept Analysis

Let X and Y be nonempty sets (of objects and attributes, respectively), let $I \subseteq X \times Y$ be a binary relation between X and Y. The triplet $\langle X, Y, I \rangle$ is called a *formal context,* the fact $\langle x, y \rangle \in I$ is interpreted as "x has y" ("y applies

to x"). A formal context corresponds to a binary data table with rows and columns corresponding to objects and attributes, respectively, such that the entry corresponding to objects x and attribute y is 1 if $\langle x, y\rangle \in I$ and 0 if $\langle x, y\rangle \notin I$. For $A \subseteq X$ and $B \subseteq Y$ we define sets $A^{\uparrow} \subseteq Y$ and $B^{\downarrow} \subseteq X$ by

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

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

Therefore, $A^{\uparrow}$ is the set of all attributes common to all objects from A and $B^{\downarrow}$ is the set of all objects common to all attributes from B. According to the traditional understanding, a pair $\langle A, B\rangle$ where $A \subseteq X$ (so-called *extent*) and $B \subseteq Y$ (so-called *intent*) is called a *formal concept of* $\langle X, Y, I\rangle$ iff $A^{\uparrow} = B$ and $B^{\downarrow} = A$. Thus, $\langle A, B\rangle$ is a formal concept of $\langle X, Y, I\rangle$ iff A is the set of all objects sharing all the attributes of B and, conversely, B is the set of all attributes common to all objects from A. Alternatively, formal concepts may be understood as maximal rectangles of the data matrix which are filled with 1's: $\langle A, B\rangle$ is a formal concept of $\langle X, Y, I\rangle$ iff it is a maximal rectangle filled with 1's which is contained in I (i.e., a maximal submatrix of $\langle X, Y, I\rangle$ filled with 1's).

The set $\mathcal{B}(X, Y, I) = \{\langle A, B\rangle \mid A^{\uparrow} = B,\ B^{\downarrow} = A\}$ is called the concept lattice induced by the input data $\langle X, Y, I\rangle$. Moreover, $\mathcal{B}(X, Y, I)$ can be equipped by a partial order relation $\leq$ defined by $\langle A, B\rangle \leq \langle C, D\rangle$ iff $A \subseteq C$ (or, equivalently, $B \supseteq D$). The partial order $\leq$ which is, in fact, a complete lattice order, models the subconcept-superconcept hierarchy: $\langle A, B\rangle \leq \langle C, D\rangle$ means that the concept $\langle C, D\rangle$ is more general than $\langle A, B\rangle$ (covers more objects, or, equivalently, less attributes).

2.2 Fuzzy Sets and Fuzzy Relations

A fuzzy set A in a universe set X [22] is a mapping assigning to each $x \in X$ a truth degree $A(x) \in L$ where L is some partially ordered set of truth degrees containing at least 0 (full falsity) and 1 (full truth). Usually, L is the unit interval $[0, 1]$ or a suitable subset of $[0, 1]$. $A(x)$ is interpreted as the degree to which x belongs to A. The notion of a fuzzy set enables us to model vaguely (nonsharply) delineated collections: For instance, the collection described linguistically as "tall men" can be modeled by a fuzzy set to which men with heights 150, 180, and 200 cm belong to degrees 0, 0.7, and 1, respectively.

In order to be able to develop the basic calculus with fuzzy sets and fuzzy relations, the set L of truth degrees needs to be equipped by suitable operations generalizing logical connectives of classical (two-valued) logic. Particularly, we will need fuzzy conjunction $\otimes$ and fuzzy implication $\rightarrow$. In the literature, there have been proposed several fuzzy conjunctions and fuzzy implications [16]. A general class of logical connectives is captured by the notion of a complete residuated lattice [1,10,14]: A complete residuated lattice is a structure $\mathbf{L} = \langle L, \wedge, \vee, \otimes, \rightarrow, 0, 1\rangle$ such that (1) $\langle L, \wedge, \vee, 0, 1\rangle$ is a complete lattice (with the least element 0, greatest element 1), i.e. a partially ordered set in which arbitrary infima ($\bigwedge$) and suprema ($\bigvee$) exist; (2) $\langle L, \otimes, 1\rangle$ is a commutative monoid, i.e. $\otimes$ is a binary operation which is commutative, associative, and $x \otimes 1 = x$ ($x \in L$); (3) $\otimes, \rightarrow$

satisfy $x \otimes y \leq z$ iff $x \leq y \to z$. Operations $\otimes$ (multiplication) and $\to$ (residuum) play the role of a fuzzy conjunction and a fuzzy implication, respectively. The class of complete residuated lattices includes structures defined on the real unit interval with $\wedge$ and $\vee$ being minimum and maximum, respectively, $\otimes$ being left-continuous t-norm, and $\to$ being its residuum, see [1,9,12,15] for details.

In what follows, $\mathbf{L}$ always refers to a complete residuated lattice and $\leq$ denotes the induced lattice order (i.e., $a \leq b$ iff $a \wedge b = a$ iff $a \vee b = b$ iff $a \to b = 1$). We write $a < b$ to denote that $a \leq b$ and $a \neq b$. Given $\mathbf{L}$, a fuzzy set with truth degrees from $\mathbf{L}$ (called also an $\mathbf{L}$-set) is a mapping $A\colon X \to L$ assigning to any $x \in X$ a truth degree $A(x) \in L$ to which x belongs to A. Similarly, a binary fuzzy relation R with truth degrees from $\mathbf{L}$ is a mapping $R\colon X \times Y \to L$ assigning to any $x \in X$ and $y \in Y$ a truth degree $R(x, y) \in L$ to which x and y are related under R. The set of all $\mathbf{L}$-sets in a universe X is denoted L^X. For a fuzzy set $A \in L^X$ and a truth degree $a \in L$ we denote by ${}^a\!/\!A$ the a-cut of A, i.e. ${}^a\!/\!A = \{x \in X \mid A(x) \geq a\}$ (the ordinary set of elements from X which belong to A to degree at least a). A fuzzy set $A \in L^X$ is called crisp if, for each $x \in X$, $A(x) \in \{0, 1\}$. Following common usage we will identify crisp fuzzy sets in X with (characteristic functions of) ordinary subsets of X. In particular, by $\emptyset$ and X we denote crisp fuzzy sets $\emptyset \in L^X$ and $X \in L^X$ such that $\emptyset(x) = 0$ and $X(x) = 1$ for each $x \in X$. For fuzzy sets $A, B \in L^X$ we put $A \subseteq B$ (A is a subset of B) if for each $x \in X$ we have $A(x) \leq B(x)$, in which case we say that A is (fully) contained in B. If for $A, B \in L^X$ we have $A \subseteq B$ and there is $x \in X$ such that $A(x) < B(x)$, we write $A \subset B$ and say that A is strictly contained in B.

2.3 Fuzzy Attributes, Fuzzy Contexts, and Formal Concepts

When dealing with real-world situations, it is very often the case that attributes that we observe on the objects of interest are fuzzy rather than bivalent. In general, an attribute y applies to an object x to some degree $I(x, y) \in L$ not necessarily being equal to 0 or 1. The larger $I(x, y)$, the more y applies to x. From the point of view of FCA, the input data table, which is no longer a binary one, is represented by a triplet $\langle X, Y, I\rangle$ (called a *formal fuzzy context*) where $I \in L^{X \times Y}$, i.e. I is a fuzzy relation between X and Y.

The agenda of formal concept analysis of data with fuzzy attributes [1,4,19] is the following. For fuzzy sets $A \in L^X$ (i.e., A is a fuzzy set of objects) and $B \in L^Y$ (i.e., B is a fuzzy set of attributes), consider fuzzy sets $A^\uparrow \in L^Y$ (fuzzy set of attributes) and $B^\downarrow \in L^X$ (fuzzy set of objects) defined by

$$A^\uparrow(y) = \bigwedge_{x \in X} (A(x) \to I(x, y)), \tag{3}$$

$$B^\downarrow(x) = \bigwedge_{y \in Y} (B(y) \to I(x, y)). \tag{4}$$

Using basic rules of fuzzy logic, one can see that $A^\uparrow(y)$ is the truth degree of "y is shared by all objects from A" and $B^\downarrow(x)$ is the truth degree of "x has all attributes from B", i.e. (3) and (4) properly generalize (1) and (2). Each $\langle A, B\rangle \in L^X \times L^Y$ such that $A^\uparrow = B$ and $B^\downarrow = A$ is called a *formal fuzzy concept of* $\langle X, Y, I\rangle$. The set of all formal fuzzy concepts of $\langle X, Y, I\rangle$ will be denoted by $\mathcal{B}(X, Y, I)$. Both the *extent* A and *intent* B of a formal fuzzy concept

$\langle A, B\rangle \in \mathcal{B}(X, Y, I)$ are fuzzy sets. This corresponds well to the intuition that a concept may apply to objects and attributes to various intermediate degrees, not necessarily to 0 and 1 only. For brevity, by $\mathrm{Int}(X, Y, I)$ we denote the set of all intents of $\langle X, Y, I\rangle$, i.e. $\mathrm{Int}(X, Y, I) = \{B \in L^Y \mid \langle A, B\rangle \in \mathcal{B}(X, Y, I) \text{ for some } A \in L^X\}$. Analogously, $\mathrm{Ext}(X, Y, I)$ denotes the set of all extents of $\langle X, Y, I\rangle$. The conceptual hierarchy in $\mathcal{B}(X, Y, I)$ is modeled by a relation $\leq$ defined on $\mathcal{B}(X, Y, I)$ by

$$\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle \quad \text{iff} \quad A_1 \subseteq A_2 \ (\text{iff } B_1 \supseteq B_2), \tag{5}$$

where $A_1 \subseteq A_2$ means that A_1 is fully contained in A_2 for each $x \in X$ (see Section 2.2). The following theorem characterizes the structure of fuzzy concept lattices.

Theorem 1 (see [4]). *The set $\mathcal{B}(X, Y, I)$ is under $\leq$ a complete lattice where the infima and suprema are given by*

$$\textstyle\bigwedge_{j\in J}\langle A_j, B_j\rangle = \big\langle \bigcap_{j\in J} A_j, (\bigcup_{j\in J} B_j)^{\downarrow\uparrow}\big\rangle, \tag{6}$$

$$\textstyle\bigvee_{j\in J}\langle A_j, B_j\rangle = \big\langle (\bigcup_{j\in J} A_j)^{\uparrow\downarrow}, \bigcap_{j\in J} B_j\big\rangle. \tag{7}$$

Moreover, an arbitrary complete lattice $\mathbf{V} = \langle V, \wedge, \vee\rangle$ is isomorphic to some $\mathcal{B}(X, Y, I)$ iff there are mappings $\gamma\colon X \times L \to V$, $\mu\colon Y \times L \to V$ such that $\gamma(X, L)$ is $\bigwedge$-dense in V; $\mu(Y, L)$ is $\bigvee$-dense in V; $a \otimes b \leq I(x, y)$ iff $\gamma(x, a) \leq \mu(y, b)$. □

If we take $\mathbf{L}$ with $L = \{0, 1\}$, i.e. there are only two truth degrees involved (our structure of truth degrees is the two-valued Boolean algebra), all notions introduced in Section 2.1 will become particular cases of notions presented in this section. This is the way the fuzzy approach generalizes the classical one [8].

Remark 1. Formal fuzzy concepts can also be characterized as maximal rectangles contained in I: For a pair $\langle A, B\rangle \in L^X \times L^Y$ (call it a rectangle), define a fuzzy relation $A \otimes B \in L^{X\times Y}$ by $(A \otimes B)(x, y) = A(x) \otimes B(y)$. $\langle A, B\rangle$ is said to be contained in I if $A \otimes B \subseteq I$. Furthermore, put $\langle A, B\rangle \sqsubseteq \langle A', B'\rangle$ iff $A \subseteq A'$ and $B \subseteq B'$. Then we have [1] that $\langle A, B\rangle$ is a formal fuzzy concept of $\langle X, Y, I\rangle$ iff it is a maximal (w.r.t. $\sqsubseteq$) rectangle contained in I.

3 Computing Fuzzy Concepts and Conceptual Hierarchy

The best known algorithm for computing formal concepts is probably Ganter's NextIntent algorithm, see [7,8]. The original NextIntent generates in a lexical order all concepts present in a (classical/bivalent) context. Graded extension of Ganter's algorithm has been presented in [3]. In this section we develop an algorithm for computing fuzzy concepts (along with their hierarchy) which is inspired by the Lindig's NextNeighbor algorithm [18].

Our goal is the following. Given an input data table represented by a formal fuzzy context $\langle X, Y, I\rangle$, (i) generate all formal fuzzy concepts $\langle A, B\rangle$ of the fuzzy concept lattice $\mathcal{B}(X, Y, I)$ and, at the same time; (ii) compute for each fuzzy concept $\langle A, B\rangle$ a set of its direct subconcepts and direct superconcepts. The sets of (direct) subconcepts/superconcepts fully determine the whole hierarchical structure of fuzzy concepts. Information about direct subconcepts and superconcepts

is of crucial importance for applications. For instance, it allows us to navigate users through the concepts, it is used as an input for geometrical method for drawing concept lattices, and more (see [8] for other applications).

Each fuzzy concept $\langle A, B\rangle$ is uniquely given by each of its components: by its extent A (since $B = A^{\uparrow}$) or by its intent B (since $A = B^{\downarrow}$). Therefore, in order to generate all fuzzy concepts from $\mathcal{B}(X, Y, I)$, it is sufficient to generate all intents from $\mathrm{Int}(X, Y, I)$ (or, equivalently, all extents). Moreover, $B \in L^Y$ is an intent from $\mathrm{Int}(X, Y, I)$ iff $B = B^{\downarrow\uparrow}$, i.e. iff B is a fixed point of the fuzzy closure operator ${}^{\downarrow\uparrow}\colon L^Y \to L^Y$ which is a composition of operators ${}^{\downarrow}$ and ${}^{\uparrow}$ defined by (3) and (4), see [1]. Thus, the task to compute all fuzzy concepts present in $\langle X, Y, I\rangle$ can be reduced to a task of computing all fixed points of a fuzzy closure operator ${}^{\downarrow\uparrow} : L^Y \to L^Y$. In what follows we forget about the operators ${}^{\downarrow}, {}^{\uparrow}$ for a while and develop the algorithm so that it accepts a general fuzzy closure operator $\mathrm{C}\colon L^Y \to L^Y$ (i.e., C is extensive, monotone, and idempotent [1]) as its input, and produces a set of its fixed points (with their hierarchy) as its output. The set of all fixed points of $\mathrm{C}\colon L^Y \to L^Y$ will be denoted by $\mathrm{fix}(\mathrm{C})$, i.e.

$$\mathrm{fix}(\mathrm{C}) = \{B \in L^Y \mid B = \mathrm{C}(B)\} = \{\mathrm{C}(B) \mid B \in L^Y\}. \tag{8}$$

Note that due to computational reasons, we restrict ourselves to finite structures of truth degrees, and always assume that X (set of objects) and Y (set of attributes) are finite. This will ensure that the set of all fuzzy concepts extracted from the data will be finite and thus enumerable in finitely many steps. For simplicity, we describe only the case when $\mathbf{L}$ is linearly ordered (the case of general finite $\mathbf{L}$ is technically more complicated and will be discussed in a full version of this paper). In the rest of the paper $\mathrm{C}\colon L^Y \to L^Y$ always denotes a fuzzy closure operator.

For convenience, denote $L = \{a_1, \ldots, a_k\}$ so that $a_1 < a_2 < \cdots < a_k$. If $i < k$, we write a_i^+ instead of a_{i+1}. Upper neighbors can be introduced as follows: $D \in \mathrm{fix}(\mathrm{C})$ is called an *upper neighbor of* $B \in \mathrm{fix}(\mathrm{C})$ (*w.r.t.* C), written $B \prec^{\mathrm{C}} D$, if (i) $B \subset D$, and (ii) there is no $D' \in \mathrm{fix}(\mathrm{C})$ such that $B \subset D' \subset D$.

Lower neighbors can be defined dually. Note that if C is ${}^{\downarrow\uparrow}$, then upper neighbors of an intent B with respect to ${}^{\downarrow\uparrow}$ are exactly the intents of the direct subconcepts of $\langle B^{\downarrow}, B\rangle$. One should not be mislead here: even if the upper neighbors of B are intents which are greater than B, they determine *subconcepts* due to the fact that greater fuzzy sets of attributes are shared by smaller fuzzy sets of objects, see the definition of conceptual hierarchy (5).

For each $B \in L^Y$ and $y \in Y$ such that $B(y) < 1$, let $\mathrm{C}\big(B \cup \{{}^{B(y)^+}\!/_y\}\big)$ be abbreviated by $[y]_B^{\mathrm{C}}$. From now on, if we write $[y]_B^{\mathrm{C}}$ we tacitly assume that $B(y) < 1$. If $[y]_B^{\mathrm{C}}$ is an upper neighbor of B w.r.t. C, then $[y]_B^{\mathrm{C}}$ will be called an upper neighbor generated by y; y is called a generator of $[y]_B^{\mathrm{C}}$. For technical reasons, we assume $Y = \{y_1, \ldots, y_n\}$ and consider a fixed order of attributes from Y given by the indices, i.e. $y_i < y_j$ iff $i < j$. In the sequel, we write just $i < j$ to denote $y_i < y_j$. The following assertion says that each upper neighbor is in fact an upper neighbor generated by some y. In addition the generator y can be chosen so that it is the greatest generator with respect to the ordering of attributes.

Lemma 1. *The following are true for any fuzzy closure operator* $\mathrm{C}\colon L^Y \to L^Y$.

(i) *For each* $B \in L^Y$ *and* $y \in Y$ *such that* $B(y) < 1$, *we have* $B(y) < ([y]^{\mathrm{C}}_B)(y)$.
(ii) *Let* $B, D \in \mathrm{fix(C)}$ *such that* $B \subset D$ *and put*

$$i = \max\{j \mid B(y_j) < D(y_j)\}. \tag{9}$$

Then, for each $k > i$, $D(y_k) = B(y_k)$. *Moreover, if* $B \prec^{\mathrm{C}} D$ *then* $D = [y_i]^{\mathrm{C}}_B$.

Proof. (i) is obvious. In order to prove (ii), let $B, D \in \mathrm{fix(C)}$ such that $B \subset D$. Clearly, $\{j \mid B(y_j) < D(y_j)\}$ is a nonempty finite set, i.e. it has a maximum. Denote the maximum by i as in (9). Take any $k > i$. Since $B \subset D$ and i is the maximum of all indices such that $B(y_j) < D(y_j)$, we get $B(y_k) = D(y_k)$. Suppose we have $B \prec^{\mathrm{C}} D$. From $B(y_i) < B(y_i)^+ \leq D(y_i)$, we get $B \cup \{{}^{B(y_i)^+}\!/_{y_i}\} \subseteq D$, which further gives $[y_i]^{\mathrm{C}}_B = \mathrm{C}\big(B \cup \{{}^{B(y_i)^+}\!/_{y_i}\}\big) \subseteq \mathrm{C}(D) = D$ because D is a fixed point of C. Furthermore, from $B \subset [y_i]^{\mathrm{C}}_B \subseteq D$ it follows that $[y_i]^{\mathrm{C}}_B = D$ because $[y_i]^{\mathrm{C}}_B \in \mathrm{fix(C)}$ and D was supposed to be an upper neighbor of B. □

Indices defined by (9) will play an important role. Therefore, we will introduce the following notation. For $\mathrm{C}\colon L^Y \to L^Y$, $B, D \in \mathrm{fix(C)}$ such that $B \subset D$, and i given by (9), we denote $y_i \in Y$ by $y^{\mathrm{C}}_B(D)$. Furthermore, we put

$$\mathcal{M}^{\mathrm{C}}_B = \{y \in Y \mid B \prec^{\mathrm{C}} [y]^{\mathrm{C}}_B \text{ and } y = y^{\mathrm{C}}_B([y]^{\mathrm{C}}_B)\}. \tag{10}$$

Note that y^{C}_B and, consequently, $\mathcal{M}^{\mathrm{C}}_B$ depend on the chosen ordering of attributes. Since the ordering is fixed, we will not mention it explicitly. Regardless of the ordering, we have that $\{[y]^{\mathrm{C}}_B \mid y \in \mathcal{M}^{\mathrm{C}}_B\}$ is a set of all upper neighbors of B w.r.t. C. Moreover, the attributes from $\mathcal{M}^{\mathrm{C}}_B$ uniquely correspond to the upper neighbors from $\{[y]^{\mathrm{C}}_B \mid y \in \mathcal{M}^{\mathrm{C}}_B\}$. This follows directly from (10) and from Lemma 1 (ii). Hence, in order to compute the upper neighbors it suffices to determine $\mathcal{M}^{\mathrm{C}}_B$. The following assertions provides us with a quick test of presence of an attribute in $\mathcal{M}^{\mathrm{C}}_B$.

Theorem 2. *Let* $B \in \mathrm{fix(C)}$ *and* $y_i \in Y$ *such that* $y_i = y^{\mathrm{C}}_B([y_i]^{\mathrm{C}}_B)$. *Then we have* $y_i \in \mathcal{M}^{\mathrm{C}}_B$ *iff for each* $y_k \in \mathcal{M}^{\mathrm{C}}_B$ *such that* $k < i$ *we have* $([y_i]^{\mathrm{C}}_B)(y_k) = B(y_k)$.

Proof. Take $B \in \mathrm{fix(C)}$ and let $y_i \in Y$ such that $y_i = y^{\mathrm{C}}_B([y_i]^{\mathrm{C}}_B)$. Hence, from (9) it follows that $([y_i]^{\mathrm{C}}_B)(y_i) > B(y_i)$ and $([y_i]^{\mathrm{C}}_B)(y_k) = B(y_k)$ $(k > i)$.

"⇒": Let $y_i \in \mathcal{M}^{\mathrm{C}}_B$, i.e. $B \prec^{\mathrm{C}} [y_i]^{\mathrm{C}}_B$. Take any $y_k \in \mathcal{M}^{\mathrm{C}}_B$ such that $k < i$. Suppose, by contradiction, that $([y_i]^{\mathrm{C}}_B)(y_k) > B(y_k)$. Since we have

$$B(y_k) < B(y_k)^+ \leq ([y_i]^{\mathrm{C}}_B)(y_k),$$

we get $B \cup \{{}^{B(y_k)^+}\!/_{y_k}\} \subseteq [y_i]^{\mathrm{C}}_B$, i.e.

$$B \subset [y_k]^{\mathrm{C}}_B = \mathrm{C}\big(B \cup \{{}^{B(y_k)^+}\!/_{y_k}\}\big) \subseteq \mathrm{C}\big([y_i]^{\mathrm{C}}_B\big) = [y_i]^{\mathrm{C}}_B. \tag{11}$$

Since $B \prec^{\mathrm{C}} [y_i]^{\mathrm{C}}_B$, (11) yields $[y_k]^{\mathrm{C}}_B = [y_i]^{\mathrm{C}}_B$. From $y_k = y^{\mathrm{C}}_B([y_k]^{\mathrm{C}}_B)$ and $k < i$ is follows that $([y_k]^{\mathrm{C}}_B)(y_i) = B(y_i)$, which is a contradiction to $([y_k]^{\mathrm{C}}_B)(y_i) = ([y_i]^{\mathrm{C}}_B)(y_i) > B(y_i)$. Therefore, $([y_i]^{\mathrm{C}}_B)(y_k) = B(y_k)$.

"⇐": Conversely, let $([y_i]^{\mathrm{C}}_B)(y_k) = B(y_k)$ be true for each $y_k \in \mathcal{M}^{\mathrm{C}}_B$ such that $k < i$. We prove that $y_i \in \mathcal{M}^{\mathrm{C}}_B$. In order to prove this claim, it suffices to check

Algorithm 1. (Compute all upper neighbors of B w. r. t. C)

```
1  procedure NEIGHBORS (B, C):
2    U := ∅
3    Min := {y ∈ Y | B(y) < 1}
4    for each y ∈ Y such that B(y) < 1:
5      D := [y]_B^C
6      Increased := {z ∈ Y | z ≠ y and B(z) < D(z)}
7      if Min ∩ Increased = ∅:
8        add D to U
9      else:
10       remove y from Min
11   return U
```

that no $[y]_B^{\mathrm{C}}$ where $y_i \neq y \in \mathcal{M}_B^{\mathrm{C}}$ is contained in $[y_i]_B^{\mathrm{C}}$ because this will give $B \prec^{\mathrm{C}} [y_i]_B^{\mathrm{C}}$ from which the claim follows immediately. Thus, take $y_i \neq y \in \mathcal{M}_B^{\mathrm{C}}$. If $y = y_k$ where $k < i$ then, by assumption, $([y_i]_B^{\mathrm{C}})(y_k) = B(y_k)$ which directly gives that $[y_k]_B^{\mathrm{C}}$ cannot be contained in $[y_i]_B^{\mathrm{C}}$ because $([y_k]_B^{\mathrm{C}})(y_k) > B(y_k) = ([y_i]_B^{\mathrm{C}})(y_k)$. If $i < k$, we have $([y_i]_B^{\mathrm{C}})(y_k) = B(y_k)$ on account of $y_B^{\mathrm{C}}([y_i]_B^{\mathrm{C}}) = y_i$. Hence, again, $[y_k]_B^{\mathrm{C}}$ cannot be contained in $[y_i]_B^{\mathrm{C}}$ which proves $y_i \in \mathcal{M}_B^{\mathrm{C}}$. □

Theorem 2 leads to Algorithm 1 for computing all upper neighbors. The algorithm accepts $\mathrm{C}\colon L^Y \to L^Y$ and $B \in \mathrm{fix}(\mathrm{C})$ as its input and produces a set of all upper neighbors of B w.r.t C.

Theorem 3. *Algorithm 1 is correct.*

Proof. The algorithm uses the following variables: $\mathcal{U}$ is a set of upper neighbors which is initially empty; *Min* is an ordinary set of attributes which should be understood as a set of possible generators of upper neighbors. The roles of $\mathcal{U}$ and *Min* are the following. The set *Min* is initially set to $\{y \in Y \mid B(y) < 1\}$ and at the end of computation, we will have $Min = \mathcal{M}_B^{\mathrm{C}}$, i.e. *Min* will be a collection of generators of upper neighbors which is built by removing attributes which do not belong to (10), the ordering of attributes (see comments before Lemma 1) is given by the order in which the loop between lines 4–10 processes the attributes. As we can see from lines 5, 8 and 10, y is left in *Min* iff $D = [y]_B^{\mathrm{C}}$ is added to $\mathcal{U}$. The important part of the algorithm is the test present at line 7. It can be seen that $Min \cap Increased = \emptyset$ happens iff $y_B^{\mathrm{C}}(D) = y$ and for each $y_k \in Min$ that has already been processed (i.e., $k < i$), we have $D(y_k) = B(y_k)$. Thus, Theorem 2 gives that test at line 7 is successful iff D is an upper neighbor of B such that $y \in \mathcal{M}_B^{\mathrm{C}}$. Hence, y can be left in *Min* and D is added to $\mathcal{U}$ which is exactly what happens between lines 7–10. By induction, we can prove that at the end of computation, $Min = \mathcal{M}_B^{\mathrm{C}}$ and $\mathcal{U} = \{[y]_B^{\mathrm{C}} \mid y \in \mathcal{M}_B^{\mathrm{C}}\}$. The full proof is omitted due to the limited scope of this paper. □

Example 1. For illustration, take $\mathbf{L} = \langle \{0, 0.5, 1\}, \min, \max, \otimes, \to, 0, 1\rangle$ with $\otimes$ and $\to$ being Łukasiewicz operations. Consider a fuzzy context from Fig. 1 (left) and an induced closure operator C being $^{\downarrow\uparrow}$. Let $B = \{^{0.5}/c, ^{0.5}/d, e\}$. When NEIGHBORS (B, C) is invoked, the procedure goes as follows. First, *Min* is set to

Algorithm 2. (Compute all fixed points of C and their hierarchy)

```
1  procedure GenerateFrom (B):
2    while B ≠ Y:
3      B* := Neighbors (B, C)
4      𝒩 := B* − ℱ
5      for each D ∈ B*:
6        add B to D*
7        if D ∈ 𝒩:
8          add D to ℱ
9      for each D ∈ 𝒩:
10       call GenerateFrom (D)
11 procedure Lattice (C, Y):
12   ℱ := ∅
13   B := C(∅)
14   add B to ℱ
15   call GenerateFrom (B)
16   return ⟨ℱ, {B*| B ∈ ℱ}, {B*| B ∈ ℱ}⟩
```

$\{a, b, c, d\}$. Then, $a \in Y$ is processed. We get $D = [a]_B^{\mathrm{C}} = \{^{0.5}/a, ^{0.5}/b, ^{0.5}/c, d, e\}$ and *Increased* $= \{b, d\}$, i.e. a is removed from *Min*. We continue with $b \in Y$ for which $D = \{^{0.5}/a, ^{0.5}/b, ^{0.5}/c, d, e\}$ and *Increased* $= \{a, d\}$. Thus, b is also removed from *Min*. Notice that a, b were removed from *Min* although both the attributes are generators of the upper neighbor D of B. This is correct because neither of them equals $y_B^{\mathrm{C}}(D)$. In the next step, we process $c \in Y$: $D = \{^{0.5}/a, ^{0.5}/b, c, d, e\}$ and *Increased* $= \{a, b, d\}$. Again, c is removed from *Min* only this time, c is not even a generator of an upper neighbor of B. Finally, we process $d \in Y$ in which case $D = \{^{0.5}/a, ^{0.5}/b, ^{0.5}/c, d, e\}$ and *Increased* $= \{a, b\}$. Since *Min* $= \{d\}$, we add D to $\mathcal{U}$. Then $\mathcal{U}$ is returned as the result of calling Neighbors (B, C).

Now, the algorithm for computing all fixed points can be described as follows. We start with the least fixed point of C which is $\mathrm{C}(\emptyset)$ and add it to the collection of found fixed points. For each newly found fixed point we first use Neighbors from Algorithm 1 to compute its upper neighbors and then we update the information about lower neighbors (D is an upper neighbor of B iff B is a lower neighbor of D). For each upper neighbor which has not been found in previous steps, we recursively repeat the process until we arrive to Y (greatest fixed point of C). The whole procedure is summarized in Algorithm 2.

Algorithm 2 consists of two procedures: Lattice accepts a closure operator $\mathrm{C}\colon L^Y \to L^Y$ and Y as its input and initiates the recursive generation of fixed points starting with the least one. The auxiliary procedure GenerateFrom does the actual job of generating fixed points. Both the procedures use the following variables: $\mathcal{F}$ is a collection of found fixed points, for each $B \in \mathcal{F}$ we denote by B^* the set of all upper neighbors of B, and by B_* we denote the set of all lower neighbors of B. Variable $\mathcal{N}$ is local in GenerateFrom and represents fixed points that were newly found during a particular call of GenerateFrom.

Theorem 4. *Algorithm 2 is correct.*

Proof. Due to the limited scope of the paper, we present only a sketch of the proof (full proof will be presented in the full version of the paper). The crucial observation is that each fixed point of C is in GenerateFrom processed only once. This in ensured at line 10, where GenerateFrom is called only for fixed points that have not been found so far (see definition of $\mathcal{N}$ at line 4). The

	a	b	c	d	e
x_1	1	0.5	0.5	1	1
x_2	1	1	1	1	0.5
x_3	0	0	0.5	0.5	1

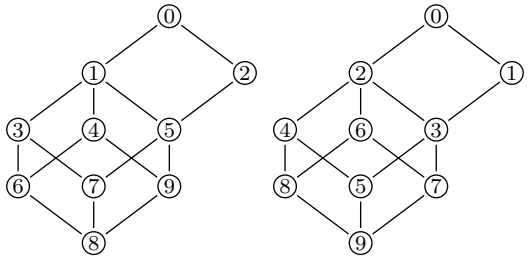

Fig. 1. Input data table and the hierarchy of conceptual clusters

information about lower neighbors (line 6) is also updated correctly because each B (considered only once) is a lower neighbor only of the fixed points which are upper neighbors of B. □

Remark 2. (a) Consider the fuzzy context from Fig. 1 (left). The corresponding structure of fuzzy concepts computed by Algorithm 2 is depicted in Fig. 1 (middle). The numbers in nodes indicate the order in which the intents determining the nodes are computed. Formals concepts corresponding to th nodes of diagram in Fig. 1 (middle) are the following:

$C_0 : \langle \{x_1, x_2, x_3\}, \{{}^{.5}/c, {}^{.5}/d, {}^{.5}/e\} \rangle$,
$C_1 : \langle \{x_1, x_2, {}^{.5}/x_3\}, \{{}^{.5}/a, {}^{.5}/b, {}^{.5}/c, d, {}^{.5}/e\} \rangle$,
$C_2 : \langle \{x_1, {}^{.5}/x_2, x_3\}, \{{}^{.5}/c, {}^{.5}/d, e\} \rangle$,
$C_3 : \langle \{x_1, x_2\}, \{a, {}^{.5}/b, {}^{.5}/c, d, {}^{.5}/e\} \rangle$,
$C_4 : \langle \{{}^{.5}/x_1, x_2, {}^{.5}/x_3\}, \{{}^{.5}/a, {}^{.5}/b, c, d, {}^{.5}/e\} \rangle$,
$C_5 : \langle \{x_1, {}^{.5}/x_2, {}^{.5}/x_3\}, \{{}^{.5}/a, {}^{.5}/b, {}^{.5}/c, d, e\} \rangle$,
$C_6 : \langle \{{}^{.5}/x_1, x_2\}, \{a, b, c, d, {}^{.5}/e\} \rangle$,
$C_7 : \langle \{x_1, {}^{.5}/x_2\}, \{a, {}^{.5}/b, {}^{.5}/c, d, e\} \rangle$,
$C_8 : \langle \{{}^{.5}/x_1, {}^{.5}/x_2\}, \{a, b, c, d, e\} \rangle$,
$C_9 : \langle \{{}^{.5}/x_1, {}^{.5}/x_2, {}^{.5}/x_3\}, \{{}^{.5}/a, {}^{.5}/b, c, d, e\} \rangle$.

Note that the order in which Algorithm 2 computes the intents (of concepts) differs from the order in which the intents are computed using the algorithm from [3]. The order in which the intents are computed in case of algorithm from [3] is depicted in Fig. 1 (right).

(b) Both the algorithms introduced in this section use a general fuzzy closure operator C instead of a fixed operator $^{\downarrow\uparrow}$ induced by a data table. The approach via arbitrary C is more general. More importantly, general closure operators play an important role for constraining the output of concept analysis, see [5], for which Algorithm 2 can also be used.

4 Comparison with Other Algorithms and Experiments

Algorithm 2 for computing fixed points of closure operators together with their hierarchy has the same asymptotic complexity as the algorithm proposed in [3]. Taking into account graded attributes, the latter claim can be proved in an analogous way as in [18]. Due to the limited scope of the paper, we omit the proof and, instead, we turn our attention to the practical performance of Algorithm 2 compared to the algorithm from [3].

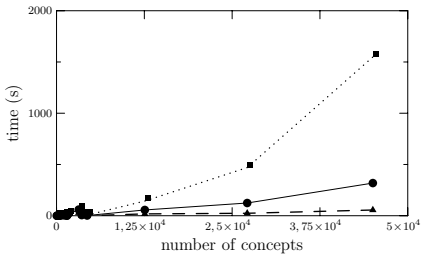

Fig. 2. Running time of algorithms for L with 5 degrees

The efficiency of the implementation of Algorithm 2 depends of the chosen data structures. The representation of $\mathcal{F}$ (and, consequently, B^* and B_* which are likely to be stored along with B) seems to be critical because the elements in $\mathcal{F}$ are frequently accessed (see line 4 of Algorithm 2). To avoid the linear time complexity of accessing elements of $\mathcal{F}$, we have organized $\mathcal{F}$ (i) as a search tree (analogously as in [18]) and (ii) by a dynamic hash table. Moreover, the sets *Min* and *Increased* in Algorithm 1 can be represented by bit arrays which significantly increases the performance (condition at line 7 of Algorithm 1 can be checked by applying the bitwise "and").

We have run several performance tests to compare Algorithm 2 to the algorithm from [3]. Since the algorithm from [3] does not compute the hierarchy of concepts, we have included in our tests an extension of that algorithm which computes the hierarchy after all the concepts are generated. Computing the complete hierarchy has asymptotic time complexity $O(n^2)$, where n is the number of concepts. The algorithms were implemented in ANSI C using the above-mentioned data structures (hashing tables and bit arrays). All experiments were run on otherwise idle Intel Pentium 4 (3.00 GHz CPU, 512 MB RAM).

To compare the performance of the algorithms we did series of experiments with randomly generated data tables with fuzzy attributes. As structures of truth degrees we used finite Łukasiewicz chains of varying size. We were interested in the dependency of running time of the algorithms on the number of generated concepts given by fixed points of $^{\downarrow\uparrow}$. The results of one of the experiments are depicted in Fig. 2. In this particular test we have used a five-element Łukasiewicz chain and we measured the average time needed for computing the concepts and their hierarchy. In the graph, the dashed line with triangles represents the average running time of the algorithm from [3], the dotted line with squares represents the running time of the algorithm from [3] followed by the hierarchy computation, and the solid line with circles corresponds to Algorithm 2.

We can see from the figure that the algorithm from [3] is the best of the three ones if we want to compute the concepts only. If we are interested in generating the concepts along with their hierarchy, Algorithm 2 proposed in this paper is considerably faster than the algorithm from [3] followed by the computation of the hierarchy. Tests with larger data and/or structures of truth degrees have

shown a similar tendency. These experimental results were expected and are in accordance with results presented in [18] for binary data.

References

1. Belohlavek, R.: Fuzzy Relational Systems: Foundations and Principles. Kluwer, Academic/Plenum Publishers, New York (2002)
2. Belohlavek, R.: Fuzzy Galois connections. Math. Logic Quarterly 45(4), 497–504 (1999)
3. Belohlavek, R.: Algorithms for fuzzy concept lattices. In: Proc. Fourth Int. Conf. on Recent Advances in Soft Computing. Nottingham, United Kingdom, pp. 200–205 (December 12–13, 2002)
4. Belohlavek, R.: Concept lattices and order in fuzzy logic. Annals of Pure and Applied Logic 128(1-3), 277–298 (2004)
5. Belohlavek, R., Vychodil, V.: Reducing the size of fuzzy concept lattices by fuzzy closure operators. In: Proc. SCIS & ISIS 2006, pp. 309–314. Tokyo Institute of Technology, Japan, (September 20–24, 2006) ISSN 1880–3741
6. Carpineto, C., Romano, G.: Concept Data Analysis. Theory and Applications. J. Wiley, Chichester (2004)
7. Ganter, B.: Two basic algorithms in concept analysis. FB4-Preprint No. 831, TH Darmstadt (1984)
8. Ganter, B., Wille, R.: Formal concept analysis. Mathematical Foundations. Springer, Heidelberg (1999)
9. Gerla, G.: Fuzzy Logic. Mathematical Tools for Approximate Reasoning. Kluwer, Dordrecht (2001)
10. Goguen, J.A.: The logic of inexact concepts. Synthese 18, 325–373 (1968-69)
11. Gratzer, G.A.: General Lattice Theory. Birkhauser, 2nd edn. (1998)
12. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht (1998)
13. Hájek, P.: On very true. Fuzzy Sets and Systems 124, 329–333 (2001)
14. Höhle, U.: On the fundamentals of fuzzy set theory. J. Math. Anal. Appl. 201, 786–826 (1996)
15. Klement, E.P., Mesiar, R., Pap, E.: Triangular Norms. Kluwer, Dordrecht (2000)
16. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic. Theory and Applications. Prentice-Hall, Englewood Cliffs (1995)
17. 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)
18. Lindig, C.: Fast concept analysis. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS, vol. 1867, pp. 152–161. Springer, Heidelberg (2000)
19. Pollandt, S.: Fuzzy Begriffe. Springer, Heidelberg (1997)
20. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered Sets, pp. 445–470. Reidel, Dordrecht, Boston (1982)
21. Zaki, M.: Mining non-redundant association rules. Data Mining and Knowledge Discovery 9, 223–248 (2004)
22. Zadeh, L.A.: Fuzzy sets. Inf. Control 8(3), 338–353 (1965)