Computing the Lattice of All Fixpoints of a Fuzzy Closure Operator

Radim Belohlavek, *Senior Member, IEEE*, Bernard De Baets, Jan Outrata, and Vilem Vychodil

***Abstract*—We present a fast bottom-up algorithm to compute all fixpoints of a fuzzy closure operator in a finite set over a finite chain of truth degrees, along with the partial order on the set of all fixpoints. Fuzzy closure operators appear in several areas of fuzzy logic and its applications, including formal concept analysis (FCA) that we use as a reference area of application in this paper. Several problems in FCA, such as computing all formal concepts from data with graded attributes or computing non-redundant bases of all attribute dependencies, can be reduced to the problem of computing fixpoints of particular fuzzy closure operators associated with the input data. The development of a general algorithm that is applicable, in particular, to these problems is the ultimate purpose of this paper. We present the algorithm, its theoretical foundations, and experimental evaluation.**

***Index Terms*—Algorithm, fixpoint, fuzzy closure operator, fuzzy logic.**

I. Introduction

THIS PAPER presents an algorithm to compute all fixpoints (or fixed points) of a fuzzy closure operator. In addition to the fixpoints, the algorithm computes the partial order on the set of all fixpoints. Fuzzy closure operators are particular mappings that assign fuzzy sets to fuzzy sets (see Section II). The concept of a fuzzy closure operator generalizes the concept of an ordinary closure operator, which is a fundamental concept in mathematics. General fuzzy closure operators were studied in several papers (see, e.g., [5], [15], [16], [21], [37], and [41]). Various particular fuzzy closure operators appear in several areas of fuzzy logic and its applications, including fuzzy relational equations (operators associated with a given fuzzy relational equation), fuzzy mathematical morphology, approximate reasoning, and fuzzy logic in narrow sense (syntactic and semantic consequence operators), and its applications such as fuzzy logic programming or formal concept analysis (FCA) of data with fuzzy attributes (concept-forming operators) (see [2], [3], [14], [18]–[20], [25], [26], [29], and [39]). In this paper, a particular area of interest is FCA of data with graded attributes (or fuzzy attributes) (see, e.g., [3], [8], and [40]). This area is related to fuzzy closure operators in a profound way, namely, fuzzy closure operators and related structures, such as fuzzy Galois connections [4], represent the mathematics behind FCA. Recall that a principal aim in FCA is to discover a hierarchical structure (so-called concept lattice) of particular clusters (so-called formal concepts) and a concise complete set of data dependencies (so-called attribute implications) from data. Since formal concepts are just the fixpoints of a particular fuzzy closure operator that is associated with the input data [3], the problem of computing all formal concepts reduces to the problem of computing all fixpoints of this operator. Similarly, attribute implications that form a so-called Guigues–Duquenne non-redundant basis of all attribute implications that are valid in the input data correspond, in a one-to-one manner, to easily recognizable fixpoints of a particular fuzzy closure operator. Hence, the problem of computing the Guigues–Duquenne basis reduces to the problem of computing all fixpoints of this operator (see an overview in [10]). The aforementioned facts led us to select FCA, in particular, the problem of computing all formal concepts from data with graded attributes, as a reference application area from which we obtain concrete fuzzy closure operators for an experimental evaluation of our algorithm.

The contributions of this paper are the following. First, we provide an algorithm to compute all fixpoints of a fuzzy closure operator in a finite set over a finite chain of truth degrees, along with the lattice order on the set of the fixpoints, as well as a justification of its correctness. The algorithm is an extension of the one from [36] for a setting with graded attributes. Second, we provide an experimental evaluation of the algorithm and compare its performance with a previously published algorithm [7]. Experiments that evaluate the performance of the algorithm from [7] have never been published previously. It turns out that the algorithm from [7] is slightly faster for computing the set of all fixpoints alone. However, for computing the set of all fixpoints along with the lattice order, the new algorithm is considerably faster. Third, we recall a previously published result that shows that classic algorithms for computing concept lattices can be used to compute fuzzy concept lattices by virtue of a particular transformation of input data and compare the performance of the proposed

Manuscript received February 16, 2009; revised September 15, 2009 and January 6, 2010; accepted December 1, 2009. Date of publication January 19, 2010; date of current version May 25, 2010. This work was supported in part by Kontakt 1-2006-33 under the project "Algebraic, logical and computational aspects of fuzzy relational modelling paradigms," in part by the Bilateral Scientific Cooperation Flanders – Czech Republic, in part by the Special Research Fund of Ghent University under Project 011S01106, in part by the Grant Agency of the Academy of Sciences of the Czech Republic under Grant 1ET101370417, in part by the Czech Science Foundation under Grant 201/05/0079 and Grant P103/10/1056, and in part by the institutional support, research plan MSM 6198959214.

R. Belohlavek and V. Vychodil were with the Department of Systems Science and Industrial Engineering, T. J. Watson School of Engineering and Applied Science, Binghamton University–State University of New York, Binghamton, NY 13902 USA. They are now with the Department of Computer Science, Palacky University, 771 47 Olomouc, Czech Republic (e-mail: radim.belohlavek@upol.cz; vilem.vychodil@upol.cz).

B. D. Baets is with the Department of Applied Mathematics, Biometrics, and Process Control, Ghent University, B-9000 Ghent, Belgium (e-mail: bernard.debaets@ugent.be).

J. Outrata is with the Department of Computer Science, Palacky University, 771 47 Olomouc, Czech Republic (e-mail: jan.outrata@upol.cz).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2010.2041006

algorithm with that of the classic algorithms. The experiments show that the proposed algorithm is considerably faster.

This paper is organized as follows. In Section II, we present preliminaries from fuzzy sets, fuzzy closure operators, and FCA of data with graded attributes. In Section III, we present the algorithm and prove its correctness. Section IV provides an experimental evaluation. In Section V, we demonstrate how the presented algorithm may be used in factor analysis of data with fuzzy attributes.

II. Preliminaries

This section provides basic notions from fuzzy sets, fuzzy closure operators, and FCA of data with graded attributes. More details can be found, e.g., in [26], [29], and [33] (fuzzy sets), [3] and [26] (fuzzy closure operators), and [3], [24], [40], and [42] (FCA).

A. *Fuzzy Sets*

We use the usual notion of a fuzzy set A in a universe set X [43], which is defined as a mapping that assigns a truth degree $A(x) \in L$ to each $x \in X$, where L is a suitable partially ordered set of truth degrees that contains at least 0 (falsity) and 1 (truth). Usually, L is either the real unit interval $[0, 1]$ or a suitable subset of it. $A(x)$ is interpreted as the degree to which x belongs to A. We use a more general approach in which L is a complete lattice that is equipped with operations. In particular, we use complete residuated lattices that are well known in fuzzy logic [27]–[30]. A complete residuated lattice is a structure $\mathbf{L} = \langle L, \wedge, \vee, \otimes, \rightarrow, 0, 1\rangle$ such that the following hold.

1) $\langle L, \wedge, \vee, 0, 1\rangle$ is a complete lattice (with the smallest element 0 and the 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 that is commutative, associative, and $a \otimes 1 = a$ for each $a \in L$.
3) $\otimes, \rightarrow$ form an adjoint pair, i.e., $a \otimes b \le c$ iff $a \le b \rightarrow c$ for every $a, b, c \in L$.

The operations $\otimes$ (multiplication) and $\rightarrow$ (residuum) play the role of a fuzzy conjunction and a fuzzy implication, respectively. For examples and further details, see [28], [29], and [32]. In the following, $\mathbf{L}$ denotes a complete residuated lattice, and $\le$ denotes the induced lattice order. (Let $a < b$ indicate that $a \le b$ and $a \ne b$.)

Given $\mathbf{L}$, a fuzzy set with truth degrees from $\mathbf{L}$ (which is also called an $\mathbf{L}$-set) is a mapping $A\colon X \rightarrow L$ that assigns to any $x \in X$ a truth degree $A(x) \in L$ to which x belongs to A. Similarly, a binary fuzzy relation R between X and Y with truth degrees from $\mathbf{L}$ is a mapping $R\colon X \times Y \rightarrow L$. The set of all $\mathbf{L}$-sets in a universe X is denoted by L^X. A fuzzy set $A \in L^X$ is called crisp if $A(x) \in \{0, 1\}$ for each $x \in X$. Following common usage, we identify crisp fuzzy sets in X with (characteristic functions of) ordinary subsets of X. In particular, let $\emptyset$ and X denote the crisp sets $\emptyset \in L^X$ and $X \in L^X$, which are defined by $\emptyset(x) = 0$ and $X(x) = 1$ for each $x \in X$. For fuzzy sets $A, B \in L^X$, we put $A \subseteq B$ if for each $x \in X$, we have $A(x) \le B(x)$, 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 if there exists $x \in X$ such that $A(x) < B(x)$, we write $A \subset B$ and say that A is strictly contained in B.

	y_1	y_2	y_3	y_4	y_5
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

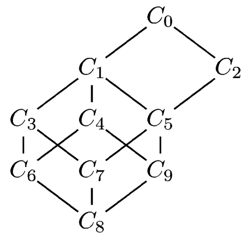


Fig. 1. Input data table and the hierarchy of conceptual clusters.

B. *Fuzzy Closure Operators*

A *fuzzy closure operator* in a universe X is a mapping $\mathrm{C} : L^X \rightarrow L^X$ that satisfies

$$A \subseteq \mathrm{C}(A) \tag{1}$$

$$A \subseteq B \text{ implies } \mathrm{C}(A) \subseteq \mathrm{C}(B) \tag{2}$$

$$\mathrm{C}(A) = \mathrm{C}(\mathrm{C}(A)) \tag{3}$$

for all fuzzy sets $A, B \in L^X$. Conditions (1)–(3) are called extensivity, monotony, and idempotency, respectively. If $A = \mathrm{C}(A)$, A is called a *fixpoint* of C. The set of all fixpoints of C is denoted by fix(C), i.e.,

$$\mathrm{fix}(\mathrm{C}) = \{A \in L^X \mid A = \mathrm{C}(A)\}.$$

C. *Formal Concept Analysis of Data With Fuzzy Attributes*

FCA provides a means to analyze data that describe objects, attributes, and their relationship. Such data are described in a table, such as in the left side of Fig. 1. The data can be represented by a triplet $\langle X, Y, I\rangle$, which is called a *formal context*, where X is a set of objects, Y is a set of attributes, and $I\colon X \times Y \rightarrow L$, i.e., I is a fuzzy relation between X and Y. For an object $x \in X$ and an attribute $y \in Y$, the degree $I(x, y)$ is interpreted as the degree to which attribute y applies to object x. For example, attribute y_2 applies to object x_1 to degree 0.5 in the data from Fig. 1.

For each formal fuzzy context $\langle X, Y, I\rangle$, we define operators $^{\uparrow}$ and $^{\downarrow}$ [3], [8], [40] as follows. 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), fuzzy sets $A^{\uparrow} \in L^Y$ (i.e., a fuzzy set of attributes) and $B^{\downarrow} \in L^X$ (i.e., a fuzzy set of objects) are defined by

$$A^{\uparrow}(y) = \bigwedge\nolimits_{x \in X} \big(A(x) \rightarrow I(x, y)\big) \tag{4}$$

$$B^{\downarrow}(x) = \bigwedge\nolimits_{y \in Y} \big(B(y) \rightarrow I(x, y)\big). \tag{5}$$

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." Every pair $\langle A, B\rangle \in L^X \times L^Y$ for which $A^{\uparrow} = B$ and $B^{\downarrow} = A$ is called a *formal* (*fuzzy*) *concept of* $\langle X, Y, I\rangle$. The set of all formal concepts of $\langle X, Y, I\rangle$ is denoted by $\mathcal{B}(X, Y, I)$ and is called a (*fuzzy*) *concept lattice* of $\langle X, Y, I\rangle$. For a formal concept $\langle A, B\rangle \in \mathcal{B}(X, Y, I)$, A and B are called the *extent* and the *intent*. For brevity, let $\mathrm{Int}(X, Y, I)$ 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 $\langle X, Y, I\rangle$.

A conceptual hierarchy in $\mathcal{B}(X, Y, I)$, which mimics the well-known subconcept–superconcept relation, is modeled by a partial order $\le$, which is defined on $\mathcal{B}(X, Y, I)$ by

$$\langle A_1, B_1\rangle \le \langle A_2\ B_2\rangle \quad \text{iff } A_1 \subseteq A_2 \quad (\text{iff } B_1 \supseteq B_2). \tag{6}$$

The following theorem describes the structure of fuzzy concept lattices.

Theorem 1 [8]: The set $\mathcal{B}(X, Y, I)$ equipped with $\le$ is a complete lattice where the infima and suprema are given by

$$\bigwedge\nolimits_{j\in J} \langle A_j, B_j\rangle = \left\langle \bigcap\nolimits_{j\in J} A_j, \left(\bigcup\nolimits_{j\in J} B_j\right)^{\downarrow\uparrow}\right\rangle \tag{7}$$

$$\bigvee\nolimits_{j\in J} \langle A_j, B_j\rangle = \left\langle \left(\bigcup\nolimits_{j\in J} A_j\right)^{\uparrow\downarrow}, \bigcap\nolimits_{j\in J} B_j\right\rangle. \tag{8}$$

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

For $\mathbf{L}$ with $L = \{0, 1\}$, i.e., the structure of truth degrees is the two-element Boolean algebra, the notions introduced previously become the ordinary notions of FCA of data with binary attributes [17], [24], [42].

Example 1: We present an illustrative example of a concept lattice of data with graded attributes. Let $\mathbf{L}$ be a three-element Łukasiewicz algebra, i.e., $\mathbf{L} = \langle\{0, 0.5, 1\}, \min, \max, \otimes, \to, 0, 1\rangle$, where $\otimes$ and $\to$ are defined by

$$a \otimes b = \max(0, a + b - 1)$$
$$a \to b = \min(1 - a + b, 1).$$

Consider a formal context $\langle X, Y, I\rangle$, which is given on the left side of Fig. 1. $X = \{x_1, x_2, x_3\}$ and $Y = \{y_1, y_2, y_3, y_4, y_5\}$ are, respectively, the sets of objects and attributes of this formal context. Table entries are degrees from $\mathbf{L}$. We now focus on formal concepts $\langle A, B\rangle$ in $\langle X, Y, I\rangle$, i.e., we focus on those $A \in L^X$ and $B \in L^Y$ for which $A^\uparrow = B$ and $B^\downarrow = A$. As can be checked, there are exactly ten formal concepts that are denoted by $C_i = \langle A_i, B_i\rangle$ ($i = 0, \ldots, 9$):

$$C_0 = \left\langle\{x_1, x_2, x_3\}, \{{}^{0.5}/y_3, {}^{0.5}/y_4, {}^{0.5}/y_5\}\right\rangle,$$
$$C_1 = \left\langle\{x_1, x_2, {}^{0.5}/x_3\},\{{}^{0.5}/y_1, {}^{0.5}/y_2, {}^{0.5}/y_3, y_4, {}^{0.5}/y_5\}\right\rangle,$$
$$C_2 = \left\langle\{x_1, {}^{0.5}/x_2, x_3\}, \{{}^{0.5}/y_3, {}^{0.5}/y_4, y_5\}\right\rangle,$$
$$C_3 = \left\langle\{x_1, x_2\}, \{y_1, {}^{0.5}/y_2, {}^{0.5}/y_3, y_4, {}^{0.5}/y_5\}\right\rangle,$$
$$C_4 = \left\langle\{{}^{0.5}/x_1, x_2, {}^{0.5}/x_3\}, \{{}^{0.5}/y_1, {}^{0.5}/y_2, y_3, y_4, {}^{0.5}/y_5\}\right\rangle,$$
$$C_5 = \left\langle\{x_1, {}^{0.5}/x_2, {}^{0.5}/x_3\}, \{{}^{0.5}/y_1, {}^{0.5}/y_2, {}^{0.5}/y_3, y_4, y_5\}\right\rangle,$$
$$C_6 = \left\langle\{{}^{0.5}/x_1, x_2\}, \{y_1, y_2, y_3, y_4, {}^{0.5}/y_5\}\right\rangle,$$
$$C_7 = \left\langle\{x_1, {}^{0.5}/x_2\}, \{y_1, {}^{0.5}/y_2, {}^{0.5}/y_3, y_4, y_5\}\right\rangle,$$
$$C_8 = \left\langle\{{}^{0.5}/x_1, {}^{0.5}/x_2\}, \{y_1, y_2, y_3, y_4, y_5\}\right\rangle,$$
$$C_9 = \left\langle\{{}^{0.5}/x_1, {}^{0.5}/x_2, {}^{0.5}/x_3\}, \{{}^{0.5}/y_1, {}^{0.5}/y_2, y_3, y_4, y_5\}\right\rangle.$$

If we order the formal concepts by their subconcept–superconcept ordering (6), we obtain a concept lattice $\mathcal{B}(X, Y, I)$, which is depicted by its Hasse diagram on the right side of Fig. 1.

Remark 1: Formal concepts can alternatively be defined as maximal rectangles that are contained in I: For a pair $\langle A, B\rangle \in L^X \times L^Y$ (which is called a rectangle), a fuzzy relation $A \otimes B \in L^{X\times Y}$ is defined by $(A \otimes B)(x, y) = A(x) \otimes B(y)$, i.e., $A \otimes B$ is the Cartesian product of A and B. $\langle A, B\rangle$ is said to be contained in I if $A \otimes B \subseteq I$. Furthermore, let $\langle A, B\rangle \sqsubseteq \langle A', B'\rangle$ iff $A \subseteq A'$ and $B \subseteq B'$. Then, $\langle A, B\rangle$ is a formal fuzzy concept of $\langle X, Y, I\rangle$ iff it is a maximal rectangle (w.r.t. $\sqsubseteq$) contained in I (see [3]).

For further information on FCA of data with fuzzy attributes, see [3], [11], [34], [38], and [40].

D. Fuzzy Closure Operators and Concept Lattices

We now describe the aforementioned connection between fuzzy closure operators and concept lattices of data with fuzzy attributes (see [3], [8], and [40] for details). It is well known that for every formal context $\langle X, Y, I\rangle$, the compound mapping ${}^{\uparrow\downarrow}\colon L^X \to L^X$ is a fuzzy closure operator in X, and ${}^{\downarrow\uparrow}\colon L^Y \to L^Y$ is a fuzzy closure operator in Y. Moreover, the fixpoints of ${}^{\uparrow\downarrow}$ are just the extents of $\langle X, Y, I\rangle$, and the fixpoints of ${}^{\downarrow\uparrow}$ are just the intents of $\langle X, Y, I\rangle$, i.e., $\mathrm{fix}({}^{\uparrow\downarrow}) = \mathrm{Ext}(X, Y, I)$, and $\mathrm{fix}({}^{\downarrow\uparrow}) = \mathrm{Int}(X, Y, I)$. Every formal 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

$$\mathcal{B}(X, Y, I) = \{\langle B^\downarrow, B\rangle \mid B \in \mathrm{Int}(X, Y, I)\}$$
$$= \{\langle B^\downarrow, B\rangle \mid B \in \mathrm{fix}({}^{\downarrow\uparrow})\}$$

which means that to compute all formal concepts of $\langle X, Y, I\rangle$, it suffices to compute all intents of $\langle X, Y, I\rangle$, i.e., all fixpoints of the fuzzy closure operator ${}^{\downarrow\uparrow}$. This also applies dually to extents. This way, the problem of computing all formal concepts of $\langle X, Y, I\rangle$ reduces to the problem of computing all fixpoints of a fuzzy closure operator that is associated with $\langle X, Y, I\rangle$.

III. Algorithm to Compute the Lattice of All Fixpoints

A. Preliminary Considerations

First, note that it is not obvious how all fixpoints of a fuzzy closure operator C can be computed efficiently because the definition provides only the following brute-force approach: Generate all $A \in L^X$ and output those for which $A = \mathrm{C}(A)$. Clearly, such an approach has an exponential time complexity and is, hence, useless.

Probably, the best-known algorithm to compute fixpoints of ordinary closure operators is Ganter's NEXTCLOSURE algorithm (see [23] and [24]). NEXTCLOSURE generates all fixpoints in a lexical order. An extension of NEXTCLOSURE for fuzzy closure operators is presented in [7], and we briefly recall this algorithm in Section IV-A.

In this section, we develop an algorithm to compute the lattice of all fixpoints of a given fuzzy closure operator, which is inspired by Lindig's NEXTNEIGHBOR algorithm [36] that computes ordinary concept lattices.

Our goal is the following. Given a fuzzy closure operator C in a set Y.

1) Compute all fixpoints of C.
2) For each fixpoint $A \in \mathrm{fix}(\mathrm{C})$, compute the sets of its direct upper and lower neighbors w.r.t. to $\subseteq$ (inclusion on fuzzy sets; see Section II-A).

Thus, our goal is to compute all fixpoints of C with the corresponding partial order. Translating this into the problem of computing concept lattices (see Section II-D), we deal with the problem of computing a concept lattice $\mathcal{B}(X, Y, I)$, along with the subconcept–superconcept ordering. The information regarding direct subconcepts and superconcepts is important for applications. For instance, it allows the navigation of users through the concept lattice. Moreover, such information is necessary for some methods to draw diagrams of concept lattices.

We assume that the set Y (universe) is finite and that L (set of truth degrees) is finite and linearly ordered (i.e., a finite chain). Namely, if either Y or L is infinite, $\mathrm{fix}(\mathrm{C})$ may also be infinite. Note that the assumption of L being linearly ordered is not essential, namely, the algorithm could easily be extended to non-linear finite lattices L by using linear extensions of the possibly non-linear order on L. However, we do not discuss this issue here since in applications, one usually uses linearly ordered L.

Also, note that neither the definition of a fuzzy closure operator nor the algorithm proposed later in this section make any use of the residuum and multiplication on L, i.e., the algorithm works with a finite chain of truth degrees and makes use of the chain ordering only. We, however, use residuated chains in the examples and experiments because we need residuum to define the fuzzy closure operators $C = {}^{\uparrow\downarrow}$, i.e., the fuzzy closure operators from FCA that we use in the examples (see Sections II-C and D).

Furthermore, we assume that for each $A \in L^X$, the closure $\mathrm{C}(A)$ of A can be computed efficiently. The complexity of computing the closures $\mathrm{C}(A)$ affects the complexity of the algorithm.

In Section III-B, we describe an algorithm that computes all upper neighbors of a given fixpoint. This algorithm is utilized by the algorithm that computes all fixpoints, which is presented in Section III-C.

B. Computing Upper Neighbors

Assume that $L = \{a_1, \ldots, a_k\}$ (set of truth degrees) such that $0 = a_1 < a_2 < \cdots < a_k = 1$. For $i < k$, we write a_i^+ instead of a_{i+1}.

Definition 1 (Upper neighbor): $D \in \mathrm{fix}(\mathrm{C})$ is called an *upper neighbor of* $B \in \mathrm{fix}(\mathrm{C})$ *with respect to* C, which is written as $B \prec^{\mathrm{C}} D$, if we have the following.

1) $B \subset D$.
2) There is no $D' \in \mathrm{fix}(\mathrm{C})$ such that $B \subset D' \subset D$.

Lower neighbors can be defined dually.

Remark 2: Note that if C is ${}^{\downarrow\uparrow}$ (cf., Section II-D), then the upper neighbors of an intent B with respect to ${}^{\downarrow\uparrow}$ are exactly the intents of the direct *subconcepts* of $\langle B^{\downarrow}, B\rangle$ because larger intents correspond to smaller concepts (see (6)). Dually, if C is ${}^{\uparrow\downarrow}$, then the upper neighbors of an extent A are exactly the extents of the direct *superconcepts* of $\langle A, A^{\uparrow}\rangle$.

For brevity, we introduce the following notation.

Notation 1: For each $B \in L^Y$ and $y \in Y$ such that $B(y) < 1$, let $[y]_B^{\mathrm{C}}$ denote $\mathrm{C}\big(B \cup \{{}^{B(y)^+}\!/y\}\big)$. From now onward, if we write $[y]_B^{\mathrm{C}}$, we tacitly assume that $B(y) < 1$.

Definition 2 (Generators of upper neighbors): If $[y]_B^{\mathrm{C}}$ is an upper neighbor of B w.r.t. C, then $[y]_B^{\mathrm{C}}$ is called an *upper neighbor generated by* y; y is called a *generator of* $[y]_B^{\mathrm{C}}$.

Example 2: In general, $B \subseteq [y]_B^{\mathrm{C}}$, but $[y]_B^{\mathrm{C}}$ need not be an upper neighbor of $B \in \mathrm{fix}(C)$. There can exist a $B' \in \mathrm{fix}(C)$ such that $B \subset B' \subset [y]_B^{\mathrm{C}}$. Indeed, consider the context on the left side of Fig. 1, and let $\mathbf{L}$ be the three-element Łukasiewicz algebra. Assume that C is ${}^{\downarrow\uparrow}$. For $B = \{{}^{0.5}\!/y_3, {}^{0.5}\!/y_4, y_5\}$, we have $[y_1]_B^{\mathrm{C}} = \{{}^{0.5}\!/y_1, {}^{0.5}\!/y_2, {}^{0.5}\!/y_3, y_4, y_5\}$, and $[y_3]_B^{\mathrm{C}} = \{{}^{0.5}\!/y_1, {}^{0.5}\!/y_2, y_3, y_4, y_5\}$. Hence, in this case, $B \subset [y_1]_B^{\mathrm{C}} \subset [y_3]_B^{\mathrm{C}}$, i.e., $[y_3]_B^{\mathrm{C}}$ is not an upper neighbor of B.

The following notation fixes an ordering on Y. The order in which the upper neighbors are computed depends on this ordering.

Notation 2: Assume that $Y = \{y_1, \ldots, y_n\}$, and consider a fixed ordering of attributes from Y given by the indexes, i.e., $y_i < y_j$ if $i < j$.

The following assertion says that each upper neighbor is, in fact, an upper neighbor that is generated by some y. In addition, the generator y can be chosen such that it is the greatest generator with respect to the ordering of attributes from Notation 2.

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

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

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

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

Proof: Point 1) immediately follows from $B(y) < B(y)^+ = \big(B \cup \{{}^{B(y)^+}\!/y\}\big)(y) \leq \mathrm{C}\big(B \cup \{{}^{B(y)^+}\!/y\}\big)(y) = ([y]_B^{\mathrm{C}})(y)$.

To prove 2), let $B, D \in \mathrm{fix}(\mathrm{C})$ such that $B \subset D$. The first claim of 2) follows directly from (9) and from $B \subset D$. To check the second claim of 2), suppose $B \prec^{\mathrm{C}} D$. By (9), $B(y_i) < D(y_i)$; hence, $B(y_i)^+ \leq D(y_i)$. Now, $B(y_i) < B(y_i)^+ \leq D(y_i)$ implies $B \cup \{{}^{B(y_i)^+}\!/y_i\} \subseteq D$, which yields $[y_i]_B^{\mathrm{C}} = \mathrm{C}\big(B \cup \{{}^{B(y_i)^+}\!/y_i\}\big) \subseteq \mathrm{C}(D) = D$ because D is a fixpoint of C. Furthermore, from $B \subset [y_i]_B^{\mathrm{C}} \subseteq D$, it follows that

$[y_i]^{\mathrm{C}}_B = D$ because $[y_i]^{\mathrm{C}}_B \in \mathrm{fix}(\mathrm{C})$, and D, by assumption, is an upper neighbor of B. ∎

As shown by Lemma 1, the indexes that are defined by (9) play an important role. Therefore, we introduce the following notation.

Notation 3: For $\mathrm{C}\colon L^Y \to L^Y$, $B, D \in \mathrm{fix}(\mathrm{C})$ such that $B \subset D$, and i given by (9), let $y^{\mathrm{C}}_B(D)$ denote $y_i \in Y$.

Theorem 2: Let $\mathrm{C}\colon L^Y \to L^Y$ be a fuzzy closure operator and $B \in \mathrm{fix}(C)$. Moreover, let $\mathcal{M}^{\mathrm{C}}_B \subseteq Y$ be defined by

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

Then, $\{[y]^{\mathrm{C}}_B \mid y \in \mathcal{M}^{\mathrm{C}}_B\}$ is the set of all upper neighbors of B.

Proof: Note that y^{C}_B and, consequently, $\mathcal{M}^{\mathrm{C}}_B$ depend on the ordering of Y that is introduced in Notation 2. By point 2) of Lemma 1, $\mathcal{M}^{\mathrm{C}}_B$ consists of the maximal elements of Y that generate the upper neighbors $[y]^{\mathrm{C}}_B$ of B (cf., (9) and Notation 2). Clearly, $\{[y]^{\mathrm{C}}_B \mid y \in \mathcal{M}^{\mathrm{C}}_B\}$ is then the set of all the upper neighbors of B. ∎

Hence, in order to compute the upper neighbors, it suffices to determine $\mathcal{M}^{\mathrm{C}}_B$. The following theorem provides a quick test of membership in $\mathcal{M}^{\mathrm{C}}_B$.

Theorem 3: Let $B \in \mathrm{fix}(\mathrm{C})$ 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: Let $B \in \mathrm{fix}(\mathrm{C})$, and let $y_i \in Y$ such that $y_i = y^{\mathrm{C}}_B([y_i]^{\mathrm{C}}_B)$. Then, (9) implies $([y_i]^{\mathrm{C}}_B)(y_i) > B(y_i)$, 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, $([y_i]^{\mathrm{C}}_B)(y_k) > B(y_k)$. Since we then 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.,

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

Since $y_i \in \mathcal{M}^{\mathrm{C}}_B$ and $y_k \in \mathcal{M}^{\mathrm{C}}_B$, we have $B \prec^{\mathrm{C}} [y_i]^{\mathrm{C}}_B$ and $B \prec^{\mathrm{C}} [y_k]^{\mathrm{C}}_B$. Then, (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 $k < i$, it 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 $y_i \in \mathcal{M}^{\mathrm{C}}_B$. To prove this claim, it suffices to check that no $[y]^{\mathrm{C}}_B$, where $y_i \neq y \in \mathcal{M}^{\mathrm{C}}_B$, is contained in $[y_i]^{\mathrm{C}}_B$ because this yields $B \prec^{\mathrm{C}} [y_i]^{\mathrm{C}}_B$ from which the claim follows evidently. Thus, $y_i \neq y \in \mathcal{M}^{\mathrm{C}}_B$. If $y = y_k$, where $k < i$, then, by assumption of the "⇐" part of Theorem 3, $([y_i]^{\mathrm{C}}_B)(y_k) = B(y_k)$. Thus, $[y_k]^{\mathrm{C}}_B$ cannot be contained in $[y_i]^{\mathrm{C}}_B$ because

$$([y_k]^{\mathrm{C}}_B)(y_k) > B(y_k) = ([y_i]^{\mathrm{C}}_B)(y_k).$$

If $i < k$, we have $([y_i]^{\mathrm{C}}_B)(y_k) = B(y_k)$ on account of $y^{\mathrm{C}}_B([y_i]^{\mathrm{C}}_B) = y_i$. Hence, again, $[y_k]^{\mathrm{C}}_B$ cannot be contained in $[y_i]^{\mathrm{C}}_B$, which proves that $y_i \in \mathcal{M}^{\mathrm{C}}_B$. ∎

Theorem 3 leads to procedure NEIGHBORS that computes all upper neighbors. The procedure accepts $\mathrm{C}\colon L^Y \to L^Y$ and $B \in \mathrm{fix}(\mathrm{C})$ as its inputs, and it outputs the set of all upper neighbors of B.

Procedure NEIGHBORS(B, C)

```
1  $\mathcal{U} := \emptyset$;
2  Min := $\{y \in Y \mid B(y) < 1\}$;
3  foreach $y \in Y$ such that $B(y) < 1$ do
4      $D := [y]^{\mathrm{C}}_B$;
5      Increased := $\{z \in Y \mid z \neq y$ and $B(z) < D(z)\}$;
6      if Min $\cap$ Increased $= \emptyset$ then
7          add $D$ to $\mathcal{U}$
8      else
9          remove $y$ from Min
10     end
11 end
12 return $\mathcal{U}$
```

Theorem 4: Procedure NEIGHBORS is correct, i.e., for a given fuzzy closure operator $\mathrm{C}\colon L^Y \to L^Y$ in a finite nonempty set Y and a fuzzy set $B \in \mathrm{fix}(\mathrm{C})$, it stops after finitely many steps and returns the set of all upper neighbors of B.

Proof: Note that Y is an arbitrary set of attributes. In order to apply previous observations, we have to introduce an explicit linear order on the attributes from Y (see Notation 2). We may assume that the ordering of attributes is given by the order in which the loop between lines 3 and 11 processes the attributes, i.e., $z < y$ will denote that $z \in Y$ is processed in the body of the loop before $y \in Y$ is processed in that loop. Let us note that the order on Y plays only a technical role for the proof and has no influence on the set of computed concepts.

The algorithm uses the following variables: $\mathcal{U}$ is a set of upper neighbors that is initially empty, and *Min* is an ordinary set of elements from Y that is to be understood as a set of possible generators of upper neighbors of B. The roles of $\mathcal{U}$ and *Min* are the following. At the beginning, *Min* is set to $\{y \in Y \mid B(y) < 1\}$. During the computation, from *Min*, we remove those elements y that are not generators of upper neighbors of B. At the end of the computation, *Min* $= \mathcal{M}^{\mathrm{C}}_B$, i.e., only generators of upper neighbors of B are left in *Min*. As we can see from lines 4, 7, and 9, y is left in *Min* iff $D = [y]^{\mathrm{C}}_B$ is added to $\mathcal{U}$. Furthermore, *Increased* is a set of elements of Y that are distinct from the currently processed elements y, which belong to $D = [y]^{\mathrm{C}}_B$ to a strictly greater degree than to B. The important part of the algorithm is the test present in line 6. By mathematical induction, it can be shown that during each cycle of the main loop, *Min* can be seen as a union of $M_y = \{z \in \mathcal{M}^{\mathrm{C}}_B \mid z < y\}$, and $N_y = \{z \in Y \mid y \leq z\}$, where y is the currently processed element. Observe that *Min* $\cap$ *Increased* $= \emptyset$ is true iff $(M_y \cup N_y) \cap$ *Increased* $= \emptyset$ iff $(M_y \cap$ *Increased*$) \cup (N_y \cap$ *Increased*$) = \emptyset$ iff both $M_y \cap$ *Increased* $= \emptyset$ and $N_y \cap$ *Increased* $= \emptyset$, which is true iff 1) for each $z \in$ *Min*, which has already been processed (i.e., $z < y$), we have $D(z) = B(z)$; and 2) $y^{\mathrm{C}}_B(D) = y$. Thus, Theorem 3 proves that the test in line 6 is successful iff $D = [y]^{\mathrm{C}}_B$ is an upper neighbor of B such that $y \in \mathcal{M}^{\mathrm{C}}_B$. Hence, y can be left in *Min*, and D is added to $\mathcal{U}$, which occurs between lines 6 and 10. Hence, by induction, we have shown that at the end of computation, *Min* $= \mathcal{M}^{\mathrm{C}}_B$, and

$\mathcal{U} = \{[y]^{\mathrm{C}}_B \mid y \in \mathcal{M}^{\mathrm{C}}_B\}$. Theorem 2, thus, yields that $\mathcal{U}$ is the set of all upper neighbors of B. ■

Example 3: Let us return to Example 1. Consider the formal context from the left side of Fig. 1, and let C be ${}^{\downarrow\uparrow}$. Assume that our structure of truth degrees is the three-element Łukasiewicz algebra. Let $B = \{{}^{0.5}/y_3, {}^{0.5}/y_4, y_5\}$. When NEIGHBORS (B, C) is invoked, the procedure runs as follows. First, *Min* is set to $\{y_1, y_2, y_3, y_4\}$. Then, $y_1 \in Y$ is processed. We get $D = [y_1]^{\mathrm{C}}_B = \{{}^{0.5}/y_1, {}^{0.5}/y_2, {}^{0.5}/y_3, y_4, y_5\}$ and *Increased* $= \{y_2, y_4\}$, i.e., y_1 is removed from *Min*. We continue with $y_2 \in Y$ for which $D = \{{}^{0.5}/y_1, {}^{0.5}/y_2, {}^{0.5}/y_3, y_4, y_5\}$, and *Increased* $= \{y_1, y_4\}$. Thus, y_2 is also removed from *Min*. Note that y_1, y_2 were removed from *Min*, although both the attributes are generators of the upper neighbor D of B. This is correct because none of them equals $y^{\mathrm{C}}_B(D)$. In the next step, we process $y_3 \in Y$: $D = \{{}^{0.5}/y_1, {}^{0.5}/y_2, y_3, y_4, y_5\}$, and *Increased* $= \{y_1, y_2, y_4\}$. Again, y_3 is removed from *Min* only this time; y_3 is not even a generator of an upper neighbor of B. Finally, we process $y_4 \in Y$, in which case, $D = \{{}^{0.5}/y_1, {}^{0.5}/y_2, {}^{0.5}/y_3, y_4, y_5\}$, and *Increased* $= \{y_1, y_2\}$. Since *Min* $= \{y_4\}$, we add D to $\mathcal{U}$. Then, $\mathcal{U}$ is returned as the result of NEIGHBORS (B, C).

C. Computing All Fixpoints

Now, the algorithm to compute all fixpoints can be described as follows. We start with the least fixpoint of C, which is $\mathrm{C}(\emptyset)$, i.e., the closure of the empty set, and add it to the collection of found fixpoints. For each newly added fixpoint, we first use NEIGHBORS 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 that has not been found in previous steps, we recursively repeat the process until we arrive at Y, which is the greatest fixpoint of C.

The whole procedure can be summarized by two separate procedures: Procedure LATTICE accepts a fuzzy closure operator $\mathrm{C}\colon L^Y \to L^Y$ and Y as its input and initiates the recursive generation of fixpoints starting with the least one. The auxiliary procedure GENERATEFROM does the actual job of generating fixpoints. Both the procedures use the following variables: $\mathcal{F}$ is a collection of fixpoints that are computed in the previous steps, B^* is the set of all upper neighbors of B, and B_* is the set of all lower neighbors of B. $\mathcal{N}$ is a local variable in GENERATEFROM, and it represents fixpoints that were newly found during a particular recursive call of the procedure.

Theorem 5: Procedure LATTICE is correct, i.e., for a given set Y and a fuzzy closure operator $\mathrm{C}\colon L^Y \to L^Y$, it stops after finitely many steps and returns the collection of all fixpoints of C, together with the sets of upper and lower neighbors for each of the fixpoints of C.

Proof: Observe that each fixpoint of C is processed only once in GENERATEFROM. This is ensured in line 11 of the procedure, where GENERATEFROM is called only for those fixpoints that have not been found so far (see definition of $\mathcal{N}$ in line 3). The information about lower neighbors (line 5) is also updated correctly because each B (which is considered only once) is a lower neighbor only of those fixpoints that are upper neighbors of B. The fact that the computation stops after finitely many steps is also evident because GENERATEFROM is called exactly once for each fixpoint. ■

```
Procedure GenerateFrom($B$)
1  while $B \neq Y$ do
2      $B^*$ := Neighbors $(B, \mathrm{C})$;
3      $\mathcal{N}$ := $B^* - \mathcal{F}$;
4      foreach $D \in B^*$ do
5          add $B$ to $D_*$;
6          if $D \in \mathcal{N}$ then
7              add $D$ to $\mathcal{F}$;
8          end
9      end
10     foreach $D \in \mathcal{N}$ do
11         GenerateFrom $(D)$
12     end
13 end
14 return
```

```
Procedure Lattice($\mathrm{C}, Y$)
1  $\mathcal{F}$ := $\emptyset$;
2  $B$ := $\mathrm{C}(\emptyset)$;
3  add $B$ to $\mathcal{F}$;
4  GenerateFrom $(B)$;
5  return $\langle \mathcal{F}, \{B^* \mid B \in \mathcal{F}\}, \{B_* \mid B \in \mathcal{F}\}\rangle$
```

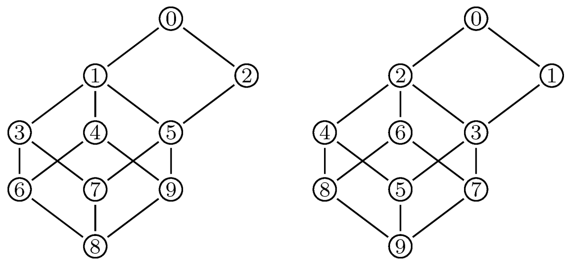


Fig. 2. Order of formal concepts computed by (left) GENERATEFROM and (right) the algorithm from [7].

Remark 3: Now, consider the computation of concept lattices, i.e., we consider $\mathrm{C} = {}^{\downarrow\uparrow}$ (cf., Section II-D). The order in which GENERATEFROM computes the intents (of formal concepts) differs from the order in which the intents are computed by the algorithm from [7]. For instance, the two algorithms compute the formal concepts from Example 1 in two distinct orders that are depicted in Fig. 2. The nodes in the diagrams are labeled by numbers that indicate the order in which the nodes are computed. The left side of Fig. 2 corresponds to GENERATEFROM, and the right side of Fig. 2 corresponds to the algorithm from [7].

IV. Experimental Evaluation

The algorithm that is presented in Section III computes fixpoints of fuzzy closure operators together with their hierarchy with a time delay (we refer to the literature on computational complexity for the concept of a time-delay complexity, e.g., [31]) that depends on the complexity of computing the closure $\mathrm{C}(A)$. For instance, for $\mathrm{C} = {}^{\downarrow\uparrow}$, the algorithm has a polynomial time-delay complexity. The arguments for this claim are the same as in case of the algorithm that is described in [7]. In this section, we mainly focus on a practical performance of our algorithm and compare it with the performance of the algorithm from [7], which we briefly review in Section IV-A.

The efficiency of implementation of our algorithm depends on the chosen data structures. The representation of $\mathcal{F}$, which is shared in the procedures GENERATEFROM and LATTICE (and, consequently, B^* and B_*, which are to be stored along with B), is critical because the elements in $\mathcal{F}$ are frequently accessed (see line 3 of procedure GENERATEFROM). To avoid a linear time complexity of accessing the elements of $\mathcal{F}$, we have organized $\mathcal{F}$ 1) as a search tree (analogously as in [36]) and 2) by a dynamic hash table. Moreover, the sets *Min* and *Increased* in procedure NEIGHBORS can be represented by bit arrays, which significantly increases the performance. (The condition in line 6 of procedure NEIGHBORS can be checked by applying the bitwise AND.)

The algorithms used for the following experiments were implemented in ANSI C using the aforementioned data structures (hash tables and bit arrays). All experiments were run on an otherwise idle Intel Pentium IV (3.00 GHz CPU and 512 MB RAM).

A. Comparison With NEXTCLOSURE *for Graded Attributes*

We have run several performance tests to compare our algorithm with the algorithm from [7] by computing concept lattices using $\mathrm{C} = {}^{\downarrow\uparrow}$ (cf., Section II-D). Since the algorithm from [7] does not compute the hierarchy of formal concepts, in our tests, we have included an extension of this algorithm that computes the hierarchy after computing all the concepts. Computing the complete hierarchy has asymptotic time complexity $O(n^2)$, where n is the number of formal concepts.

For the sake of completeness, we briefly review the algorithm from [7]. The algorithm is a natural extension of the well-known ordinary NEXTCLOSURE [24]. It computes all formal concepts of a fuzzy concept lattice $\mathcal{B}(X, Y, I)$ in a lexicographic order of the intents. It starts with $\langle \emptyset^{\downarrow}, \emptyset^{\downarrow\uparrow} \rangle$ (which is the lexicographically smallest formal concept when ordered by intents). Then, for the currently computed formal concept $\langle A, B \rangle$, the algorithm uses an efficient test to compute its lexicographic successor $\langle B^{+\downarrow}, B^{+} \rangle$. This step is repeated until the algorithm reaches the last formal concept $\langle Y^{\downarrow}, Y \rangle$.

We performed a series of experiments with randomly generated data tables with graded attributes. The fill ratio of the tables, i.e., the percentage of nonzero entries in the table, ranged from 5% to 90%, and the size of the tables was up to 1000×20 (objects $\times$ attributes). As the structures of truth degrees, we used finite BL-chains [29] (e.g., finite linearly ordered structures with Łukasiewicz and minimum operations) of varying size up to 101 truth degrees, i.e., up to $L = \{0, 0.01, 0.02, \ldots, 0.98, 0.99, 1\}$.

Our observations are summarized by several graphs in which the dashed lines represent average running times of the algorithm from [7] (i.e., NEXTCLOSURE), the dotted lines show average running times of the algorithm from [7] followed by computing the hierarchy of concepts, and the solid lines correspond to the algorithm from Section III.

TABLE I
RUNNING TIME OF ALGORITHMS FOR L WITH FIVE DEGREES (ŁUKASIEWICZ)

concepts	35	76	176	218	366	523	709	1076
our algorithm (ms)	4	12	20	20	40	102	122	310
NEXTCLOSURE (ms)	4	8	12	16	34	62	94	228
NEXTCLOSURE + order (ms)	0	10	28	24	66	174	250	1012

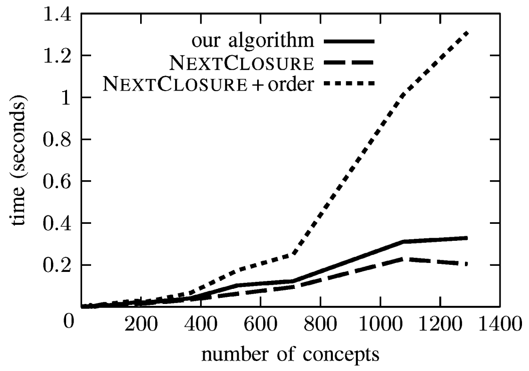


Fig. 3. Dependence of running time of algorithms on the size of concept lattice.

In all experiments, we observed the running times of the algorithms and their dependency on various characteristics, such as the size of input data tables (formal contexts), distributions of truth degrees in data tables, numbers of truth degrees in data tables, and the like. Some of the observations are summarized as follows.

The first experiment studies a dependency of the running time on the size of concept lattices (which is the number of formal concepts, i.e., fixpoints). The results of a typical experiment are depicted in Table I and Fig. 3. Table I shows the running times in milliseconds. Note that Łukasiewicz chains were used as structures of truth degrees. Fig. 3 demonstrates the following tendency: With growing sizes of concept lattices, our algorithm significantly outperforms graded NEXTCLOSURE followed by computing the hierarchy of concepts. The graded NEXTCLOSURE alone (without computing the hierarchy) is usually faster than our algorithm.

Another series of experiments focused on the influence of the fill ratio, i.e., the fraction of nonzero table entries in the table. The results are shown in Fig. 4. The figure shows that the differences in performance of the algorithms are not significant for sparse data tables, i.e., data tables where most entries are zeros. If the data tables become dense, our algorithm is several times faster than graded NEXTCLOSURE algorithm followed by the computation of hierarchy.

In all the previous tests, we used Łukasiewicz chains as structures of truth degrees. The experiments in which we used different structures of truth degrees have shown similar average behavior. For instance, Fig. 5 shows the dependency of the running time on the number of formal concepts if we use structures of truth degrees with minimum conjunction (Gödel structures) instead of the Łukasiewicz one.

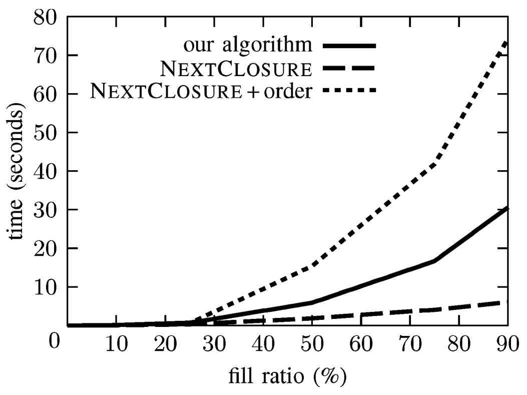


Fig. 4. Running time of algorithms for data tables 500×20.

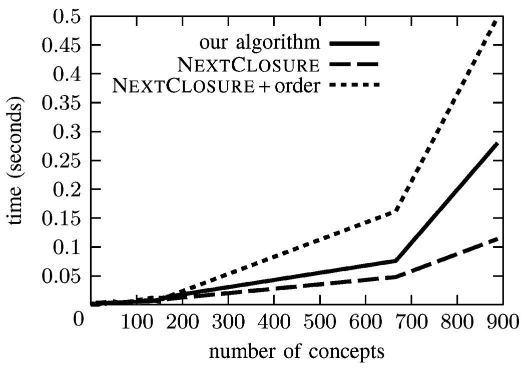


Fig. 5. Running time of algorithms for L with minimum conjunction (Gödel).

We can see from the figures in this section that the algorithm from [7] is the best, given that we want to compute only formal concepts. If we wish to compute the lattice order as well (this is necessary if we want to depict the resulting hierarchy by its Hasse diagram), the algorithm that is proposed in this paper is considerably faster than graded NEXTCLOSURE followed by the computation of hierarchy.

B. Computing Fuzzy Concept Lattices Using Algorithms for Ordinary Concept Lattices

It is well known that every fuzzy concept lattice is isomorphic to an ordinary concept lattice [6], [40], namely, a fuzzy concept lattice $\mathcal{B}(X,Y,I)$ is isomorphic to an ordinary concept lattice $\mathcal{B}(X \times L, Y \times L, I^{\times})$, where the incidence relation $I^{\times}$ between sets $X \times L$ and $Y \times L$ is defined by $\langle\langle x,a\rangle,\langle y,b\rangle\rangle \in I^{\times}$ iff $a \otimes b \leq I(x,y)$. As a result, to compute a fuzzy-concept lattice $\mathcal{B}(X,Y,I)$, the classic algorithms may be used to compute ordinary concept lattices as follows: First, $\langle X,Y,I\rangle$ is transformed to $\langle X \times L, Y \times L, I^{\times}\rangle$; second, a classic algorithm is applied to compute $\mathcal{B}(X \times L, Y \times L, I^{\times})$ (and then, $\mathcal{B}(X,Y,I)$ is obtained by a straightforward transformation [6], [40]).

A question that we answer in this section is how the performance of the algorithm described in Section III can be compared with the performance of the earlier procedure in which one first transforms the data are transformed first, and then, classic algorithms are applied to compute ordinary concept lattices. For this purpose, we use Lindig's NEXTNEIGHBOR [36], which is one of

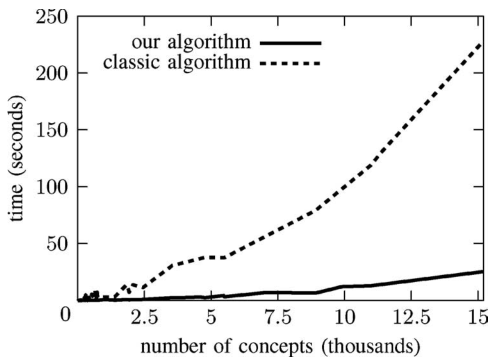


Fig. 6. Average running time of (solid line) our algorithm using fuzzy closure operators that are induced by graded contexts and (dotted line) the classic algorithm [36] using closure operators that are induced by corresponding bivalent contexts. The graph shows time dependency on the size of concept lattice for L with Łukasiewicz operations.

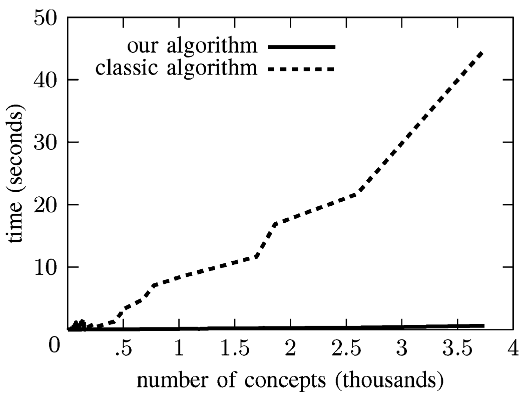


Fig. 7. Average running time of (solid line) our algorithm using fuzzy closure operators that are induced by graded contexts and (dotted line) the classic algorithm [36] using closure operators that are induced by the corresponding bivalent contexts. The graph shows time dependency on the size of concept lattice for L with minimum (Gödel) operations.

the fastest and most widely used classic algorithms that compute ordinary concept lattices (i.e., sets of all formal concepts with the lattice order) (see [35]). In addition, our algorithm is an extension of classic Lindig's algorithm for graded attributes.

We performed a series of experiments with randomly generated data tables with graded attributes. In these experiments, the fill ratio of the tables ranged from 5% to 33%, and the size of the tables was up to 50×20. The finite linearly ordered structures with Łukasiewicz and minimum (Gödel) operations, which are used as the structures of truth degrees, were of varying size up to 21 truth degrees.

As in the experiments described in the previous section, we observed the running times of the computations and their dependency on various characteristics. The results that show the dependency of the running time on the size of concept lattices are depicted in Figs. 6 and 7 for Łukasiewicz operations and minimum operations, respectively. In the graphs, the dotted lines show average running times of the computation on transformed data with bivalent attributes, and the solid lines correspond to the computation on the original data with graded attributes.

We can see that the computation of the concept lattices of bivalent data, which is transformed from graded data, is very expensive. Our algorithm is considerably faster, especially in the case of the structures of truth degrees with minimum operations (see Fig. 7). This shows the usefulness of our algorithm, i.e., an algorithm that is designed directly for data with fuzzy attributes.

V. Application

In this section, we present an application of the algorithm from Section III in the area of exploratory data analysis, namely, in factor analysis of data with fuzzy attributes. The main aim in factor analysis (see, e.g., [1]) is to extract possibly a small number of factors from a data matrix that describes objects and their attributes, and provide a model that explains the data in terms of the discovered factors. In the case of real-valued attributes, classic factor analysis and its various generalizations proved to be useful. In the case of binary data, various methods have been proposed (see, e.g., [22] and references therein). It has been proved in [13] that formal concepts associated with the input binary matrix are optimal factors for Boolean factor analysis and an efficient approximation algorithm is developed for Boolean factor analysis based on the optimality result. The case of binary data was extended to data with fuzzy attributes in [9] and [12]. In the following, we describe the factor analysis of data with fuzzy attributes and show how the algorithm from Section III can be used to extract factors from such data. We also include an illustrative example.

The input data consist of an $n \times m$ matrix that describes n objects (matrix rows) and m attributes (matrix columns). A matrix entry that corresponds to row i and column j is interpreted as the degree to which attribute j applies to object i. As mentioned previously, the degrees are considered to belong to some complete residuated lattice $\mathbf{L}$. Clearly, such a matrix can be identified with a binary fuzzy relation I between $X = \{1, \ldots, n\}$ (objects) and $Y = \{1, \ldots, m\}$ (attributes). The aim of factor analysis is to decompose I into a product $I = A \circ B$ of an $n \times k$ object-factor matrix A and a $k \times m$ factor-matrix B, with the number k of factors as small as possible. $A \circ B$ is the sup-t-norm product, i.e., $(A \circ B)_{ij} = \bigvee_{l=1}^{k} A_{il} \otimes B_{lj}$. The new k factors revealed by such a decomposition can be interpreted as new (more general or substantial) variables of which the original m attributes are particular manifestations. A_{il} and B_{lj} are interpreted as the degrees to which factor l applies to object i and to which attribute j is a particular manifestation of factor l, respectively.

The following theorem, which can be regarded as an optimality theorem that shows that formal concepts from the concept lattice $\mathcal{B}(X, Y, I)$ of I are optimal factors for such decompositions, was proven in [9].

Theorem 6: If $I = A \circ B$ for $n \times k$ and $k \times m$ binary matrices A and B, there exists a set $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ of formal concepts of I with $|\mathcal{F}| \leq k$ such that for the $n \times |\mathcal{F}|$ and $|\mathcal{F}| \times m$ matrices $A_{\mathcal{F}}$ and $B_{\mathcal{F}}$, we have $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$.

In this theorem, the matrices $A_{\mathcal{F}}$ and $B_{\mathcal{F}}$ are constructed from $\mathcal{F}$ as follows. The columns of $A_{\mathcal{F}}$ are the extents of the formal concepts from $\mathcal{F}$, and the rows of $B_{\mathcal{F}}$ are the intents of the formal concepts from $\mathcal{F}$. That is for $\mathcal{F} = \{\langle C_1, D_1\rangle, \ldots, \langle C_k, D_k\rangle\}$, we put $(A_{\mathcal{F}})_{il} = C_l(i)$, and $(B_{\mathcal{F}})_{lj} = D_l(j)$. Thus, the concept lattice $\mathcal{B}(X, Y, I)$ that is associated with the input matrix I plays the role of a space of optimal factors (formal concepts) that are hierarchically ordered by the subconcept–superconcept relation $\leq$.

In [12], the authors developed a greedy approximation algorithm of computing the optimal decompositions $I = A \circ B$. The algorithm works as follows. In step 1, the concept lattice $\mathcal{B}(X, Y, I)$ is computed from the input matrix I. In step 2, a particular greedy strategy is used, and the current best factor (a formal concept that "explains" the largest portion of yet "unexplained data") is selected iteratively. The algorithm continues until all data are "explained" by the selected factors. This way, the algorithm selects a small set $\mathcal{F}$ of factors from $\mathcal{B}(X, Y, I)$ for which $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$.

The algorithm that is described in Section III (cf., Section II-D and Remark 3) can be employed in step 1. The subconcept–superconcept hierarchy $\leq$ on the set $\mathcal{B}(X, Y, I)$ of all formal concepts represents an important structure on the set of factors. Namely, due to $\leq$, the set of factors is not flat. Instead, it has a natural hierarchical structure that represents a useful information for a user. Moreover, during step 2, a user may be asked if the current candidate factor to be added to $\mathcal{F}$ has a good interpretation and makes sense in the particular domain. If yes, the factor is added to $\mathcal{F}$. If not, the user may either browse the neighborhood of the candidate factor in the hierarchically ordered space $\mathcal{B}(X, Y, I)$ of all factors and select another one (with a better interpretation while still explaining a large portion of unexplained data) or ask the algorithm to generate another candidate factor. In the following, we present an illustrative example of factor analysis of ordinal data that is performed by the procedure described in this paragraph.

TABLE II
2004 Olympic Games Decathlon—Scores of Top Five Athletes

	10	*lj*	*sp*	*hj*	40	11	*di*	*pv*	*ja*	15
Sebrle	894	1020	873	915	892	968	844	910	897	680
Clay	989	1050	804	859	852	958	873	880	885	668
Karpov	975	1012	847	887	968	978	905	790	671	692
Macey	885	927	835	944	863	903	836	731	715	775
Warners	947	995	758	776	911	973	741	880	669	693

Legend: 10—100 m; *lj*—long jump; *sp*—shot put; *hj*—high jump; 40—400 m; 11—110 m hurdles; *di*—discus throw; *pv*—pole vault; *ja*—javelin throw; 15—1500 m.

Table II contains the results of top five athletes in 2004 Olympic Games decathlon in points that are obtained using the International Association of Athletics Federations (IAAF) Scoring Tables for Combined Events. Note that the IAAF Scoring Tables provide us with an ordinal scale and a ranking function assigning the scale values to athletes. We are going to look at whether there are some general factors in terms of sports physiology that explain these results.

We first transform the data to a five-element scale

$$L = \{0.00, 0.25, 0.50, 0.75, 1.00\}$$

by a natural transformation and rounding. As a consequence, the factors then have a simple reading, namely, the grades to which a factor applies to an athlete can be described in natural language as "not at all," "little bit," "half," "quite," "fully," or the like. In

TABLE III
FACTOR FORMAL CONCEPTS FOR DECATHLON DATA

F_i	*Extent* C_i	*Intent* D_i
F_1	$\{^{.5}/\text{Sebrle}, \text{Clay}, \text{Karpov}, ^{.5}/\text{Macey}, ^{.75}/\text{Warners}\}$	$\{10, lj, ^{.75}/sp, ^{.75}/hj, ^{.5}/40, 11, ^{.5}/di, ^{.25}/pv, ^{.25}/ja, ^{.5}/15\}$
F_2	$\{\text{Sebrle}, ^{.75}/\text{Clay}, ^{.25}/\text{Karpov}, ^{.5}/\text{Macey}, ^{.25}/\text{Warners}\}$	$\{^{.5}/10, lj, sp, hj, ^{.75}/40, 11, ^{.75}/di, ^{.75}/pv, ja, ^{.75}/15\}$
F_3	$\{^{.75}/\text{Sebrle}, ^{.5}/\text{Clay}, ^{.75}/\text{Karpov}, \text{Macey}, ^{.5}/\text{Warners}\}$	$\{^{.5}/10, ^{.5}/lj, ^{.75}/sp, hj, ^{.75}/40, ^{.5}/11, ^{.75}/di, ^{.25}/pv, ^{.5}/ja, 15\}$
F_4	$\{\text{Sebrle}, ^{.75}/\text{Clay}, ^{.75}/\text{Karpov}, ^{.5}/\text{Macey}, \text{Warners}\}$	$\{^{.5}/10, ^{.75}/lj, ^{.5}/sp, ^{.5}/hj, ^{.75}/40, 11, ^{.25}/di, ^{.5}/pv, ^{.25}/ja, ^{.75}/15\}$
F_5	$\{^{.75}/\text{Sebrle}, ^{.75}/\text{Clay}, \text{Karpov}, ^{.75}/\text{Macey}, ^{.25}/\text{Warners}\}$	$\{^{.75}/10, ^{.75}/lj, ^{.75}/sp, ^{.75}/hj, ^{.75}/40, ^{.75}/11, di, ^{.25}/pv, ^{.25}/ja, ^{.75}/15\}$
F_6	$\{^{.75}/\text{Sebrle}, ^{.5}/\text{Clay}, \text{Karpov}, ^{.75}/\text{Macey}, ^{.75}/\text{Warners}\}$	$\{^{.75}/10, ^{.75}/lj, ^{.75}/sp, ^{.75}/hj, 40, ^{.75}/11, ^{.5}/di, ^{.25}/pv, ^{.25}/ja, ^{.75}/15\}$
F_7	$\{\text{Sebrle}, \text{Clay}, ^{.25}/\text{Karpov}, ^{.5}/\text{Macey}, ^{.25}/\text{Warners}\}$	$\{^{.5}/10, lj, ^{.75}/sp, ^{.75}/hj, ^{.5}/40, 11, ^{.75}/di, ^{.5}/pv, ja, ^{.5}/15\}\rangle$

Legend: 10—100 m; lj—long jump; sp—shot put; hj—high jump; 40—400 m; 11—110 m hurdles; di—discus throw; pv—pole vault; ja—javelin throw; 15—1500 m.

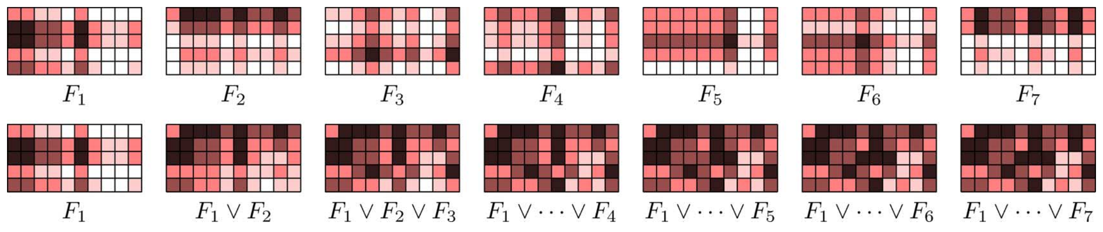
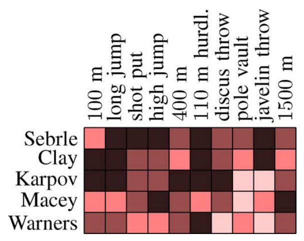
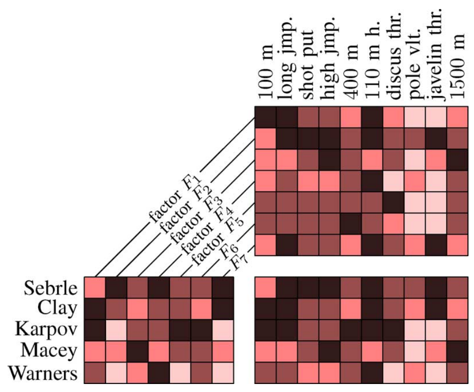

F_1
F_2
F_3
F_4
F_5
F_6
F_7
F_1
$F_1 \vee F_2$
$F_1 \vee F_2 \vee F_3$
$F_1 \vee \cdots \vee F_4$
$F_1 \vee \cdots \vee F_5$
$F_1 \vee \cdots \vee F_6$
$F_1 \vee \cdots \vee F_7$

Fig. 8. (First line) Rectangular matrices that represent the factors for decathlon data. (Second line) $\bigvee$-superposition of factors.

addition, we use the Łukasiewicz t-norm on L. The following array visually represents the resulting 5×10 matrix I:



The rows and columns correspond to the athletes and decathlon disciplines, respectively. The entries are colored boxes that represent degrees from a five-element scale $L = \{0, 0.25, 0.5, 0.75, 1\}$ (the darker the color, the larger the degree). Using the previously described algorithm, matrix I was decomposed into a product $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$ of a 5×7 matrix $A_{\mathcal{F}}$ and a 7×10 matrix $B_{\mathcal{F}}$, with a set $\mathcal{F} = \{\langle C_l, D_l \rangle \mid l = 1, \ldots, 7\}$ being a set of formal concepts of I that is described later. Note that the user always accepted the first-candidate formal concept in this case. The decomposition found by the algorithm described previously is depicted as follows:



Matrix $A_{\mathcal{F}}$ is the bottom left matrix with athletes' names labeling the rows; matrix $B_{\mathcal{F}}$ is the top matrix with disciplines' names labeling the columns. As mentioned previously, the lth column of $A_{\mathcal{F}}$ and the lth row of $B_{\mathcal{F}}$ are the vectors that correspond to the extent C_l and the intent D_l, respectively, of the lth factor $F_l = \langle C_l, D_l \rangle$ $(l = 1, \ldots, k)$. The formal concepts (factors) from $\mathcal{F}$ are depicted with a detailed description in Table III.

The first line of Fig. 8 shows the rectangular matrices that correspond to the factors from $\mathcal{F} = \{F_1, \ldots, F_7\}$. The rectangular matrix that corresponds to $F_l = \langle C_l, D_l \rangle$ is constructed as the Cartesian product of C_l and D_l, i.e., the entry at row i and column j is $C_l(i) \otimes D_l(j)$. Since $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$, the input matrix is then equal to the $\bigvee$-superposition of the rectangular matrices that correspond to the factors, i.e., $I_{ij} = \bigvee_{l=1}^{7} C_l(i) \otimes D_l(j)$ (see [9]). The second line of Fig. 8 shows the $\bigvee$-superpositions $F_1 \vee \cdots \vee F_l$ of the first l factors $(l = 1, \ldots, 7)$. As we can see from the visual inspection of the matrices, already the first two or three factors explain the data reasonably well, i.e., $F_1 \vee \cdots \vee F_3$ approximates I reasonably well.

Let us now consider the interpretation of the first three factors. For this purpose, it is convenient to inspect the rectangular matrices from the first line of Fig. 8 and, to see more details, Table III. *Factor*F_1: Manifestations of this factor with grade 1 are 100 m, long jump, and 110 m hurdles. This factor can be interpreted as the ability to run fast for short distances. Note that this factor particularly applies to Clay and Karpov, which is well known in the world of decathlon. *Factor*F_2: Manifestations of this factor with grade 1 are long jump, shot put, high jump, 110 m hurdles, and javelin. F_2 can be interpreted as the ability to apply very high force in a very short term (explosiveness). F_2 particularly applies to Sebrle and, to a lesser degree, to Clay, who are known for this ability. *Factor*F_3: Manifestations with grade 1 are high jump and 1500 m. This factor is typical for lighter, and not very muscular, athletes. (Too many muscles prevent jumping high and running long distances.) Macey, who is evidently that type among decathletes (196 cm and 98 kg), is the athlete to whom the factor applies to degree 1. These are the most important factors behind data matrix I.

VI. Conclusion

We presented an algorithm that computes all fixpoints of a given fuzzy closure operator in a finite set over a finite chain of truth degrees, along with the partial order on the fixpoints. The algorithm is applicable to general fuzzy closure operators. It overcomes the exponential time complexity, which is implicitly present in the definition of a fixpoint, by computing the partially ordered set of fixpoints in a bottom-up manner by employing efficient tests. We proved correctness of these tests and the algorithm. Particular fuzzy closure operators appear in several areas. We focused on FCA, where computing of fixpoints of various closure operators is of crucial importance. In particular, we used the problem of computing a concept lattice from data with fuzzy attributes for experimental evaluation of our algorithm.

Future research is to be directed toward the development of further algorithms for computing fixpoints of fuzzy closure operators, both general fuzzy closure operators as well as various particular closure operators. These problems were thoroughly studied in the binary setting, i.e., for ordinary closure operators (see, e.g., [35]). For fuzzy closure operators, however, our paper is the first systematic contribution.

References

[1] D. J. Bartholomew and M. Knott, *Latent Variable Models and Factor Analysis*, 2nd ed. London, U.K.: E. Arnold, 1999.

[2] W. Bandler and L. Kohout, "Special properties, closures and interiors of crisp and fuzzy relations," *Fuzzy Sets Syst.*, vol. 26, no. 3, pp. 317–331, 1988.

[3] R. Belohlavek, *Fuzzy Relational Systems: Foundations and Principles*. New York: Kluwer/Plenum, 2002.

[4] R. Belohlavek, "Fuzzy Galois connections," *Math. Logic Q.*, vol. 45, no. 4, pp. 497–504, 1999.

[5] R. Belohlavek, "Fuzzy closure operators," *J. Math. Anal. Appl.*, vol. 262, pp. 473–489, 2001.

[6] R. Belohlavek, "Reduction and a simple proof of characterization of fuzzy concept lattices," *Fundam. Inf.*, vol. 46, no. 4, pp. 277–285, 2001.

[7] R. Belohlavek, "Algorithms for fuzzy concept lattices," in *Proc. 4th Int. Conf. Recent Adv. Soft Comput.*, Nottingham, U.K., Dec. 12–13, 2002, pp. 200–205.

[8] R. Belohlavek, "Concept lattices and order in fuzzy logic," *Ann. Pure Appl. Logic*, vol. 128, no. 1–3, pp. 277–298, 2004.

[9] R. Belohlavek, "Optimal decompositions of matrices with grades," in *Proc. IEEE Int. Conf. Intell. Syst.*, Varna, Bulgaria, 2008, pp. 15-2–15-7, IEEE Catalog No. CFP08802-PRT, ISBN 978-1-4244-1740-7.

[10] R. Belohlavek and V. Vychodil, "Attribute implications in a fuzzy setting," in *Proc. ICFCA* (Lecture Notes in Artificial Intelligence, vol. 3874), R. Missaoui and J. Schmid, Eds. Berlin, Germany: Springer-Verlag, 2006, pp. 45–60.

[11] R. Belohlavek and V. Vychodil, "Reducing the size of fuzzy concept lattices by fuzzy closure operators," in *Proc. SCIS, ISIS*, Tokyo, Japan, Sep. 20–24, 2006, pp. 309–314, ISSN 1880-3741.

[12] R. Belohlavek and V. Vychodil, "Factor analysis of incidence data via novel decomposition of matrices," in *Proc. Int. Conf. Formal Concept Anal.* (Lecture Notes in Computer Science, vol. 5548), 2009, pp. 83–97.

[13] R. Belohlavek and V. Vychodil, "Discovery of optimal factors in binary data via a novel method of matrix decomposition," *J. Comput. Syst. Sci.*, vol. 76, pp. 3–20, 2010. DOI: 10.1016/j.jcss.2009.05.002.

[14] L. Biacino and G. Gerla, "Closure systems and L-subalgebras," *Inf. Sci.*, vol. 33, pp. 181–195, 1984.

[15] L. Biacino and G. Gerla, "An extension principle for closure operators," *J. Math. Anal. Appl.*, vol. 198, pp. 1–24, 1996.

[16] U. Bodenhofer, "A unified framework of opening and closure operators with respect to arbitrary fuzzy relations," *Soft Comput.*, vol. 7, no. 4, pp. 220–227, 2003.

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

[18] B. De Baets, E. Kerre, and M. Gupta, "The fundamentals of fuzzy mathematical morphology. Part 2: Idempotence, convexity and decomposition," *Int. J. Gen. Syst.*, vol. 23, pp. 307–322, 1995.

[19] B. De Baets, "Generalized idempotence in fuzzy mathematical morphology," in *Fuzzy Techniques in Image Processing* (Studies in Fuzziness and Soft Computing, vol. 52), E. Kerre and M. Nachtegael, Eds. Heidelberg, Germany: Physica-Verlag, 2000, pp. 58–75.

[20] B. De Baets, "Analytical solution methods for fuzzy relational equations," in *Fundamentals of Fuzzy Sets* (The Handbooks of Fuzzy Sets Series, vol. 1), D. Dubois and H. Prade, Eds. Norwell, MA: Kluwer, 2000, ch. 6, pp. 291–340.

[21] W. Flüshöh and U. Höhle, "L-fuzzy contiguity relations and L-fuzzy closure operators in the case of completely distributive, complete lattices L," *Math. Nachrichten*, vol. 145, no. 1, pp. 119–134, 1990.

[22] A. A. Frolov, D. Húsek, I. P. Muraviev, and P. A. Polyakov, "Boolean factor analysis by Hopfield-like autoassociative memory," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 698–707, May 2007.

[23] B. Ganter, "Two basic algorithms in concept analysis," Technische Hoschule Darmstadt, Darmstadt, Germany, Tech. Rep., FB4-Preprint no. 831, 1984.

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

[25] G. Gerla, "Graded consequence relations and fuzzy closure operators," *J. Appl. Non-Classical Logics*, vol. 6, pp. 369–379, 1996.

[26] G. Gerla, *Fuzzy Logic. Mathematical Tools for Approximate Reasoning*. Dordrecht, The Netherlands: Kluwer, 2001.

[27] J. A. Goguen, "The logic of inexact concepts," *Synthese*, vol. 18, no. 3/4, pp. 325–373, 1968.

[28] S. Gottwald, *A Treatise on Many-Valued Logics*. Baldock, U.K.: Research Studies, 2001.

[29] P. Hájek, *Metamathematics of Fuzzy Logic*. Dordrecht, The Netherlands: Kluwer, 1998.

[30] U. Höhle, "On the fundamentals of fuzzy set theory," *J. Math. Anal. Appl.*, vol. 201, pp. 786–826, 1996.

[31] D. S. Johnson, M. Yannakakis, and C. H. Papadimitrou, "On generating all maximal independent sets," *Inf. Process. Lett.*, vol. 27, pp. 129–133, 1988.

[32] E. P. Klement, R. Mesiar, and E. Pap, *Triangular Norms*. Dordrecht, The Netherlands: Kluwer, 2000.

[33] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic. Theory and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[34] S. Krajči. (2005). A generalized concept lattice. *Logic J. IGPL* [Online]. *13(5)*, pp. 543–550. Available: http://dx.doi:10.1093/jigpal/jzi045

[35] S. O. Kuznetsov and S. A. Obiedkov, "Comparing performance of algorithms for generating concept lattices," *J. Exp. Theor. Artif. Intell.*, vol. 14, no. 2/3, pp. 189–216, 2002.

[36] C. Lindig, "Fast concept analysis," in *Working With Conceptual Structures—Contributions to ICCS 2000*, G. Stumme, Ed. Aachen, Germany: Shaker-Verlag, 2000, pp. 152–161.

[37] A. S. Mashour and M. H. Ghanim, "Fuzzy closure spaces," *J. Math. Anal. Appl.*, vol. 106, pp. 154–170, 1985.

[38] J. Medina, M. Ojeda-Aciego, and J. Ruiz-Calviño. (2009). Formal concept analysis via multi-adjoint concept lattices. *Fuzzy Sets Syst.* [Online]. *160*, pp. 130–144. Available: http://dx.doi.org/10.1016/j.fss.2008.05.004

[39] V. Murali, "Lattice of fuzzy subalgebras and closure systems in I^X," *Fuzzy Sets Syst.*, vol. 41, pp. 101–111, 1991.

[40] S. Pollandt, *Fuzzy Begriffe*. Berlin, Germany: Springer-Verlag, 1997.

[41] R. O. Rodríguez, F. Esteva, P. Garcia, and L. Godo, "On implicative closure operators in approximate reasoning," *Int. J. Approx. Reason.*, vol. 33, pp. 159–184, 2003.

[42] R. Wille, "Restructuring lattice theory: An approach based on hierarchies of concepts," in *Ordered Sets*, I. Rival, Ed. Dordrecht, The Netherlands: Reidel, 1982, pp. 445–470.

[43] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.

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

He was appointed a Full Professor of computer science by the President of the Czech Republic in 2005. From 2001 to 2007, he was the Head of the Department of Computer Science, Palacky University. Since 2007, he has been a Professor of systems science with Binghamton University—State University of New York, Binghamton. His current research interests include the areas of uncertainty and information, logic and algebra, fuzzy logic and fuzzy sets, and relational data analysis. He was a principal investigator of numerous grants in these areas. He has authored two monographs (Kluwer/Springer). He is also the author or coauthor of more than 130 papers presented at conferences and published in journals, including the *Journal of Computer and System Sciences*, *Annals of Pure and Applied Logic*, the *Archive for Mathematical Logic*, *Neural Computation*, the *Journal of Logic and Computation*, the *Journal of Mathematical Analysis and Applications*, *Fundamenta Informaticae*, *Fuzzy Sets and Systems*, *Information Sciences*, the *International Journal of General Systems*, the *Journal of Experimental and Theoretical Artificial Intelligence*, and *Mathematical Logic Quarterly*.

Prof. Belohlavek is a Member of the Association for Computing Machinery and the American Mathematical Society.

Bernard De Baets was born in 1966. He received the M.Sc. and Ph.D. degrees in mathematics in 1988 and 1995, respectively, and postgraduated in knowledge technology in 1991, all *summa cum laude* from Ghent University, Ghent, Belgium.

Since 2008, he has been a Full Professor of applied mathematics with Ghent University, where he leads KERMIT, the Research Unit Knowledge-Based Systems. He has authored or coauthored more than 190 papers published in international journals and nearly 50 book chapters. He is a member of the Editorial Boards of various international journals, in particular, Co-Editor-in-Chief of *Fuzzy Sets and Systems*.

Prof. De Baets coordinates EUROFUSE, which is the EURO Working Group on Fuzzy Sets, and is a member of the Board of Directors of the European Society for Fuzzy Logic and Technology, the Technical Committee on Artificial Intelligence and Expert Systems of the International Association of Science and Technology for Development, and the Administrative Board of the Belgian Operations Research (OR) Society. He was a recipient of the Government of Canada Award in 1988. In 2006, he became Honorary Professor of Budapest Tech, Hungary.

Jan Outrata received the B.Sc. degree in computer science, the M.Sc. degree in 2003, and the Ph.D. degree in mathematics in 2006, all from Palacky University, Olomouc, Czech Republic.

Since 2005, he has been with the Department of Computer Science, Palacky University. His current research interests include fuzzy logic, fuzzy relational systems, relational data analysis, clustering, and knowledge engineering. He has authored or coauthored more than 20 papers presented at conferences and published in journals, including the *Journal of Computer and System Sciences*, the *International Journal of General Systems*, and the *International Journal Foundations of Computer Science*.

Vilem Vychodil received the B.Sc. degree in computer science, the M.Sc. degree in 2002, and the Ph.D. degree in mathematics in 2004, all from Palacky University, Olomouc, Czech Republic.

From 2000 to 2007, he was with the Department of Computer Science, Palacky University. Since 2007, he has been with the Department of Systems Science and Industrial Engineering, T. J. Watson School of Engineering and Applied Science, Binghamton University—State University of New York, Binghamton. His current research interests include fuzzy logic, fuzzy relational systems, relational data analysis, uncertainty in data, mathematical logic, and logical foundations of knowledge engineering. He has authored one monograph (Springer). He is also the author or coauthor of more than 70 papers presented at conferences and published in journals, including the *Archive for Mathematical Logic*, *Mathematical Logic Quarterly*, the *Logic Journal of the Interest Group in Pure and Applied Logic*, the *Journal of Experimental and Theoretical Artificial Intelligence*, *Fuzzy Sets and Systems*, and the *Journal of Multiple-Valued Logic and Soft Computing*.

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