Data mining using L-fuzzy concept analysis

Sajal Saha

Computer Science

Submitted in partial fulfilment
of the requirements for the degree of

Master of Science

Faculty of Mathematics & Science, Brock University
St. Catharines, Ontario

Abstract

Association rules in data mining are implications between attributes of objects that hold in all instances of the given data. These rules are very useful to determine the properties of the data such as essential features of products that determine the purchase decisions of customers. Normally the data is given as binary (or crisp) tables relating objects with their attributes by yes-no entries. We propose a relational theory for generating attribute implications from many-valued contexts, i.e, where the relationship between objects and attributes is given by a range of degrees from no to yes. This degree is usually taken from a suitable lattice where the smallest element corresponds to the classical no and the greatest element corresponds to the classical yes. Previous related work handled many-valued contexts by transforming the context by scaling or by choosing a minimal degree of membership to a crisp (yes-no) context. Then the standard methods of formal concept analysis were applied to this crisp context. In our proposal, we will handle a many-valued context as is, i.e., without transforming it into a crisp one. The advantage of this approach is that we work with the original data without performing a transformation step which modifies the data in advance.

Acknowledgements

It is my pleasure to express my profound gratitude to my supervisor, Dr. Michael Winter not only for his supervision but also for the advice, motivation, and encouragement from the beginning of this research work. I am fortunate enough to have Dr. Winter as my supervisor, as his constant support, guiding and efforts to clarify concepts help me to overcome all obstacles and finish this thesis work smoothly. Without help from him, it would be impossible for me to finish this thesis work.

I would also like to thank the supervisory committee members Professor Sheridan Houghten, Professor Robson De Grande and the external reviewer Professor Michael Gruninger, who reviewed my work and provided insightful comments. The academic and financial assistance that I got from the Brock University Computer Science Department assisted me to complete my degree. I am grateful for this.

Finally, I want to thank all of my family members and friends for their support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Data mining is the process of analyzing large data set to discover hidden and interesting patterns or useful information for specific purposes. Data mining methods can be categorized as finding association rules, classification rules, clustering rules, sequential patterns, etc. and this classification is based on the classes of knowledge gained from the data set [9]. Among them, the most commonly seen data mining approach is the finding association rules or attributes implication.

An association rule or an attribute implication is a formula of the form $A \to B$ where A and B are set of attributes. This kind of rule plays a significant role in data analysis. In a Boolean context, the implication $A \to B$ means each object which has all attributes in A also has all attributes in B. As an example, let us consider a data set of customers purchasing items at a computer store. Let us furthermore assume that the data supports the attribute implication $\{laptop\} \to \{headphone,\ cooler,\ laptop\ bag\}$. This rule indicates that a customer buying a laptop will also buy a set of headphones, a cooler, and a laptop bag.

Formal Concept Analysis (FCA) is a conceptual framework based on the lattice theory for analyzing, mining, and visualizing data. It was introduced by Rudolf Wille in the mid-1980s. FCA deals with a formal context which is a binary relation between a set of objects and a set of attributes. We can interpret a formal context as a table with rows and columns respectively corresponds to the objects and attributes. The table entries contain 1 or 0 based on whether an object has or does not have the corresponding attribute. FCA can produce two outputs from the formal context: one is a lattice of formal concepts and another is a non-redundant basis of attribute implications. The basis can be used to generate all implications valid in the given set of data [11].

It might not always be clear whether an object has a certain attribute or not. Besides other sources of uncertainty, the attribute might not be clearly defined, or we are not able to measure the given object precisely enough. Therefore, we are interested in allowing fuzzy attributes, i.e., an attribute that an object may have up to a certain degree. For example, a student may know programming completely (represented by 1) or may not know anything about programming (represented by 0) or may know programming to a certain extend (represented by value 2). The truth degree by which an object has an attribute is usually taken from an appropriate lattice L of truth degrees. A typical choice for L is the real unit interval [0,1] or some subset of thereof [1].

FCA provides a way to handle a multi-valued context, i.e., a context with L different from $[0, 1]$, by conceptual scaling. Conceptual scaling transformed a many-valued context to a Boolean context in accordance with certain rules. We need to interpret the derived concepts from the one-valued context as the concepts of multi-valued context and this process is called conceptual scaling. In the scaling process, all attributes of a many-valued context is interpreted by means of a scale. Let us consider a multi-valued context shown in Table 1.1.

Table 1.1: Multi-valued context

	age	eligibility
James	20	1
George	44	0
Chelsie	80	1

Here age is a multi-valued attribute that must be scaled before applying basic FCA framework. We can discretize the multi-valued attribute age into several binary-value attributes as young(15-24 years), middle-aged(25-44 years), and old(above 44 years). We can transform Table 1.1 into a table with yes/no attributes like the one in Table 1.2.

Table 1.2: Boolean context after conceptual scaling

	young	middle-aged	old	eligibility
James	1	0	0	1
George	0	1	0	0
Chelsie	0	0	1	1

In [6], they identified the problems of scalar-valued fuzzy relationship in representing the fuzzy formal context and proposed a solution using interval-valued fuzzy formal contexts (in short IVFF context). Generally, a fuzzy formal context is assigned a membership value from the scale $L = [0, 1]$ which determines to what extent an object has an attribute. But It is difficult to precisely determine the truth value from the scale L, and this scalar-valued representation of fuzzy context also unable to handle incomplete context [4]. They identified the representation of a fuzzy formal context using sub-interval from the scale L is convenient and it is also beneficial in a handling incomplete context. The problem of their proposed work is to determine the distinct intervals of the context.

A new scaling technique has been proposed in [2], namely scaling of general attributes to fuzzy attributes, which consider a static set of the fuzzy attribute to normalize the data. Conceptual scaling transforms the general attribute, e.g. nominal, ordinal, etc., to the Boolean attribute. But this kind of scaling is very sensitive to the user's selection of scale attributes and results in a large difference in the generated concepts lattice with a small change in scale attribute. This effect can be diminished by applying fuzzy attributes in scaling instead of a crisp attribute [2]. But one of the disadvantages of their proposed scaling method is the arbitrariness of boundaries of the fuzzy attribute set.

In [5], they convert the many-valued context to fuzzy by allowing multiple relevance memberships value for each attribute. They used the concept of a fuzzy bag instead of a traditional representation of documents as a vector of terms. They attached additional information related to relevance by setting multiple membership values to a single term. As a result, the quality of the conceptual hierarchy was improved but their experimental results generated some irrelevant attribute implications. They adapted a pruning technique using threshold confidence to eliminate them.

All these examples show that scaling produces errors and/or undesired results in previous research works. The selection of the scale for the multi-valued attribute is not mathematically compelling, it is a matter of interpretation. The disadvantage of this kind of scaling is that it's not simple to decide the boundaries between scale attributes. Consequently, the experimental result will be biased by the user assumption since there is no logical explanation behind the attribute boundary. The purpose of this thesis is to construct and discuss a framework of L-fuzzy formal concept analysis in handling a multi-valued context without any data pre-processing. Previous research works focused on different scaling technique to handle a multi-valued context. The advantage of our approach is that we work with actual data without any preparatory transformation step. Therefore, our approach does not introduce any bias into the data by choosing a certain scale. We use relational algebraic formulations for generating the concepts and attribute implications from a given context. This includes a Boolean as well as a multi-valued contexts.

We developed a library using Java programming language to prove the functionality of our proposed algebraic formulas. The program can dynamically handle any context providing the corresponding lattice, source, and target information to describe the context. We can perform all the basic operations including generation of concepts and attribute implications using our implementation. There is also a way to save program output for future analysis.

We briefly describe the content of the thesis. Chapter 2 presents the minimum mathematical background information which is necessary for reading the main text of the work. The chapter is divided into four main sections, introducing lattices, relations, L-fuzzy relations, and categories of relations. Chapter 3 briefly describes Formal Concept Analysis and Fuzzy Formal Concept Analysis. We explain the background theory of concept analysis with examples in this chapter. In Chapter 4, we will focus on concept lattice in allegories and attribute implications. The main contribution of this thesis is explained in this chapter. In Chapter 5, we will discuss the implementation of our approach. The first two sections summarize the basic functions in our library. The last section describes the steps to operate the program. The last chapter includes our concluding remarks and some ideas about future work.

Chapter 2

Mathematical Preliminaries

This chapter will briefly introduce the main concepts and notations of lattice theory and set theory with examples. These concepts will be used to develop our research work. For more details we refer to [3, 7, 12, 13].

2.1 Lattices

Definition 2.1. *A partially ordered set (or poset) is a set P together with a binary relation $x \leq y$ satisfying the following properties:*

1. *For all $x \in P$, $x \leq x$ (Reflexivity),*
2. *For all $x, y, z \in P$, if $x \leq y$ and $y \leq x$, then $x = y$ (Antisymmetric),*
3. *For all $x, y, z \in P$, if $x \leq y$ and $y \leq z$, then $x \leq z$ (Transitivity).*

The symbol $\leq$ is read as "contains","includes", or "is less than or equal to".

Example 1. The set of all subsets $P(X)$ of a set X is a partially ordered set where $x \leq y$ means y includes x as a subset, i.e., for every element z, if it is an element of x then it must be an element of y. For example, consider a three element set $\{x, y, z\}$ and all its eight subset: $\varnothing$, $\{x\}$, $\{y\}$, $\{z\}$, $\{x, y\}$, $\{y, z\}$, $\{z, x\}$, $\{x, y, z\}$ with the inclusion relation ($\subseteq$) as a order relation. This is a very simple mathematical example of partially ordered set.

Definition 2.2. *Let $(L, \leq)$ is a poset and S is an arbitary subset of L, i.e., $S \subseteq L$. Then an element $l \in L$ is called an upper bound (a lower bound) of S iff $s \leq l$ ($l \leq s$) for all $s \in S$.*

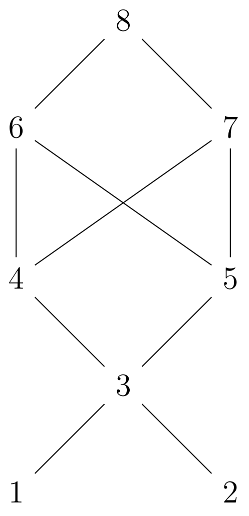


Figure 2.1: The greatest lower bound and the least upper bound of A = {6, 7} and B = {4,5}

Example 2. Figure 2.1 illustrates that upper bounds (lower bounds) not need to be unique. There may be more than one upper bounds (lower bounds). For example, the upper bounds of $\{4, 5\}$ are $\{6, 7, \text{ and } 8\}$. On the other hand, the set $\{6, 7\}$ has only one upper bound, the element $\{8\}$. The lower bounds of $\{6, 7\}$ are $\{1, 2, 3, 4, \text{ and } 5\}$ while the set $\{1, 2\}$ has no lower bound.

Definition 2.3. *Let consider $(L, \leq)$ is poset and $S \subseteq L$. The least upper bound (greatest lower bound) of S is an element $u \leq x$ $(x \leq u)$ for all upper bound (lower bound) x of S.*

Example 3. Least upper bound (greatest lower bound) may not exist but if they do, they are unique. For example, the set $\{4, 5\}$ does not have a least upper bound because the element 6 and 7 are incomparable, i.e., neither 6 nor 7 is smaller or equal than all upper bound of $\{4, 5\}$. The least upper bound of the set $\{1, 2\}$ is the element 3. The greatest lower bounds of $\{4, 5\}$ is 3 since it the greatest element among all lower bounds while the set $\{1, 2\}$ has no greatest lower bound as there is no lower bound of $\{1, 2\}$.

Definition 2.4. *An upper semilattice (lower semilattice) is a partially ordered set $(L, \leq)$ in which for each pair (x, y) of their elements, the least upper bound or join $(x \vee y)$ (the greatest lower bound or meet $(x \wedge y)$) exists. A partial order set $(L, \leq)$ is called lattice iff it is both a lower and upper semilattice.*

Definition 2.5. *An upper semilattice (lower semilattice) is called complete iff the supremum* $\bigvee X$ *(infimum* $\bigwedge X$*) exists for all subsets.*

An complete upper semilattice has a greatest element $\bigvee L$. Dually, a complete lower semilattice has a lowest element $\bigwedge L$. We can define the notion of a semilattice or lattice also algebraically. A set together with two binary $\wedge$ and $\vee$ is a lattice iff it satisfies the following properties:

1. $x \wedge x = x,$ $\quad x \vee x = x$ (idempotent),
2. $x \wedge y = y \wedge x,$ $\quad x \vee y = y \vee x$ (commutative),
3. $x \wedge (y \wedge z) = (x \wedge y) \wedge z,$ $\quad x \vee (y \vee z) = (x \vee y) \vee z$ (associative),
4. $x \wedge y = x \Longleftrightarrow x \leq y,$ $\quad x \vee y = y \Longleftrightarrow x \leq y$ (consistent).

Definition 2.6. *A bounded lattice* $(L, \wedge, \vee, 0, 1)$ *is a lattice* $(L, \wedge, \vee)$ *with least element 0 and greatest element 1.*

Definition 2.7. *A distributed lattice is a lattice* $(L, \wedge, \vee)$ *in which the meet* $(\wedge)$ *and join* $(\vee)$ *operations are distribute over each other i.e., for all* $x, y, z \in L$ we have:

1. $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z),$
2. $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z).$

2.1.1 Complete Heyting Algebra

This class of lattices is interesting by providing a notion of complement or negation. This is also known as a relative pseudo-complement.

Definition 2.8. *Heyting algbera* $(H, \wedge, \vee, \rightarrow, 0, 1)$ *is a bounded lattice* $(H, \wedge, \vee, 0, 1)$ *with an binary operation* $(x \rightarrow y)$ *of pseudo-complement* x *relative to* y *or implication which satisfy following properties for all* $x, y, z \in L$*:*

1. $z \leq y \rightarrow x \Longleftrightarrow x \wedge z \leq y.$

Theorem 1. *Complete Heyting algebra is a complete distributive lattice satisfying following properties:*

1. for all x : $x \wedge \bigvee M = \bigwedge\limits_{y \in M} (x \wedge y).$

2.2 Relations

A relation R between two sets X and Y is a set of pairs of elements from X and Y, i.e., $R \subseteq X \times Y$ where $X \times Y$ denotes the set of pairs. If R is a relation between X and Y we frequently write $R : X \to Y$.

Example 4. Consider two set F and C representing the set of fruits, and their color and taste respectively.

$$F = \{\text{Apple, Banana, Cherry, Lemon}\}$$
$$C = \{\text{Red, Green, Yellow, Sweet, Sour}\}$$

The relation between set F and C is denoted below as the subset of their Cartesian product.

$R = \{$(Apple, Red), (Apple, Green), (Apple, Sweet), (Apple, Sour), (Banana, Yellow), (Banana, Sweet), (Cherry, Red), (Cherry, Sweet), (Lemon, Yellow), (Lemon, Sour)$\}$

A Boolean matrix M with dimension $m \times n$ can be used to present the relation R between two finite set F $\{f_1, f_2, \ldots, f_m\}$ and C $\{c_1, c_2, \ldots, c_n\}$ where the entries for $i \in \{1, 2, .., m\}$ and $j \in \{1, 2, .., n\}$ are given by

$$M_{ij} = \begin{cases} 1 & \text{if } (f_i, c_j) \in \text{R} \\ 0 & \text{if } (f_i, c_j) \notin \text{R} \end{cases}$$

We can generate the equivalent Boolean matrix from the relation mentioned in previous example as follows:

$$M = \begin{array}{c|ccccc} & \textit{Red} & \textit{Green} & \textit{Yellow} & \textit{Sweet} & \textit{Sour} \\ \hline \textit{Apple} & 1 & 1 & 0 & 1 & 1 \\ \textit{Banana} & 0 & 0 & 1 & 1 & 0 \\ \textit{Cherry} & 1 & 0 & 0 & 1 & 0 \\ \textit{Lemon} & 0 & 0 & 1 & 0 & 1 \end{array}$$

If it is clear from the context we may drop the row and column labels and simply write

$$M = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

2.2.1 Basic Operations

There are some fundamental operations called union, intersection, implication, composition, conversion, complement, left-residual, right-residual, symmetric quotient on relations. In terms of matrices, these operations can be executed component-wise by using the Boolean operations AND($\wedge$), OR($\vee$), and NOT($\neg$) to the corresponding entries in the matrices.

Definition 2.9. *Given two relation $P : X \to Y$ and $Q : X \to Y$ of same type, we define few basic operations as follows:*

$$\begin{aligned}
\text{Union}, P \cup Q &:= \{(x, y) \in X \times Y \mid (x, y) \in P \vee (x, y) \in Q\}\\
\text{Intersection}, P \cap Q &:= \{(x, y) \in X \times Y \mid (x, y) \in P \wedge (x, y) \in Q\}\\
\text{Implication}, P \to Q &:= \{(x, y) \in X \times Y \mid (x, y) \notin P \vee (x, y) \in Q\}\\
\text{Conversion}, P^T &:= \{(y, x) \in Y \times X \mid (x, y) \in P\}\\
\text{Complement}, \overline{P} &:= \{(x, y) \in X \times Y \mid (x, y) \notin P\}\\
\text{Identity}, \mathbb{I} &= \{(x, x) \in X \times X \mid x \in X\}
\end{aligned}$$

Example 5. Consider P and Q as following matrices to perform the operations mentioned above

$$P = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \text{ and } Q = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

The union operation between P and Q returns following matrix

$$P \cup Q = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The intersection operation between P and Q returns following matrix

$$P \cap Q = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The implication operation between P and Q returns following matrix

$$P \to Q = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

The complementation of P will return following matrix

$$\overline{P} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

The transpose of relation P will return following matrix

$$P^T = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Definition 2.10. *The relational composition $P;Q : X \to Z$ of $P : X \to Y$ and $Q : Y \to Z$ is defined by*

$$R;S = \{(x,z) \in X \times Z \mid \exists y \in Y : (x,y) \in R \wedge (y,z) \in S\}$$

Example 6. For example, consider two relation P and Q as follow:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } Q = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The composition of P and Q will return following relation.

$$P;Q = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Definition 2.11. *The relation $\varepsilon : X \to P(X)$ is defined by $(x,y) \in \varepsilon$ iff $x \in y$.*

Example 7. For example, consider a set $X = \{x_1, x_2\}$. Then ε has the following matrix representation:

$$\epsilon = \begin{array}{c} \\ x_1 \\ x_2 \end{array} \begin{array}{cccc} \varnothing & \{x_2\} & \{x_1\} & \{x_1,x_2\} \\ \left(\begin{matrix} 0 \\ 0 \end{matrix}\right. & \begin{matrix} 0 \\ 1 \end{matrix} & \begin{matrix} 1 \\ 0 \end{matrix} & \left.\begin{matrix} 1 \\ 1 \end{matrix}\right) \end{array}$$

Definition 2.12. *The left residual $P/Q : X \to Z$ of two relations $P : X \to Y$ and $Q : Z \to Y$ is defined by*

$$P/Q = \{(x,z) \in X \times Z \mid \forall y \in Y : (z,y) \in Q \to (x,y) \in P\}$$

Example 8. For example, consider two relation P and Q as follow:

$$P = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \text{ and } Q = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

The left residual of P and Q will return following relation.

$$P/Q = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Intuitively, the matrix P/Q has a 1 at (i,j) iff the j-th row of Q is included in the i-th row of P in the sense that if Q has a 1 at (j,k), then P has a 1 at (i,k).

Definition 2.13. *The right residual $P\backslash Q : Y \to Z$ of two relations $P : X \to Y$ and $Q : X \to Z$ is defined by*

$$P\backslash Q = \{(y,z) \in Y \times Z \mid \forall x \in X : (x,y) \in P \to (x,z) \in Q\}$$

Example 9. For example, consider two relation P and Q as follow:

$$P = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \text{ and } Q = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

The right residual of P and Q ($P\backslash Q$) will return following relation.

$$P\backslash Q = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The matrix $P\backslash Q$ has a 1 at (i,j) iff the i-th column of P and the j-th column of Q are coincide in the sense that if P has a 1 at (k,i), then Q has a 1 at (k,j).

Definition 2.14. *The symmetric quotient $P(syq)Q : Y \to Z$ of two relations $P : X \to Y$ and $Q : X \to Z$ is defined by*

$$syq(P,Q) = \{(y,z) \in Y \times Z \mid \forall x \in X : (x,y) \in P \leftrightarrow (x,z) \in Q\}$$

Example 10. For example, consider two relation P and Q as follow:

$$P = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \text{ and } Q = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

The symmetric quotient between P and Q will return following relation.

$$sqy(P,Q) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

We also need the projections from pair of set to the set. The projection operations can be performed in two ways, one from the Cartesian product of two set to the first component and another is to the second component which are called Pi(π) and Rho(ρ) respectively.
We can define π and ρ as follow considering two sets X and Y:

$$\pi : X \times Y \to X \qquad \rho : X \times Y \to Y$$

For example, if $X = \{x_1,\ x_2\}$ and $Y = \{y_1,\ y_2\}$ are two classical set then π and ρ can be defined as follow

$$\pi = \begin{array}{c} \\ (x_1,y_1) \\ (x_1,y_2) \\ (x_2,y_1) \\ (x_2,y_2) \end{array} \!\!\begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \end{array} \qquad \rho = \begin{array}{c} \\ (x_1,y_1) \\ (x_1,y_2) \\ (x_2,y_1) \\ (x_2,y_2) \end{array} \!\!\begin{array}{c} \begin{array}{cc} y_1 & y_2 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \end{array}$$

2.3 L-Fuzzy Relation

Fuzzy relations generalize classical relations by weighing the relationship between two objects with a degree. For example, a fuzzy relation “friend” indicates the degree of friendship between two person while classical relation describe whether two person either being friend or not.

Definition 2.15. *Let A and B are two sets and L a complete Heyting algbera. Then we can define a L-fuzzy subset A' of A as a function $A' : A \to L$. An L-fuzzy relation between A and B is a function $R : A \times B \to L$.*

Fuzzy sets and relations can be represented using a two-dimensional matrix of membership value between objects. Consider two set $A = \{a_1, a_2, ..., a_m\}$ and $B = \{b_1, b_2, ..., b_n\}$. Then an $m \times n$ matrix of membership degree as follow can be define to represent a fuzzy relation R.

$$\mathcal{R} = \begin{array}{c} \\ a_1 \\ a_2 \\ ... \\ a_m \end{array} \begin{array}{c} \begin{array}{cccc} b_1 & b_2 & ... & b_n \end{array} \\ \begin{pmatrix} R(a_1,b_1) & R(a_1,b_2) & ... & R(a_1,b_n) \\ R(a_2,b_1) & R(a_2,b_2) & ... & R(a_2,b_n) \\ ... & ... & ... & ... \\ R(a_m,b_1) & R(a_m,b_2) & ... & R(a_m,b_n) \end{pmatrix} \end{array}$$

For example, consider the sets $A = \{a_1, a_2, a_3\}$ and $B = \{b_1, b_2, b_3, b_4\}$. Then the following L is a complete Heyting algebra and the matrix is an L-fuzzy relation between A and B.

$$R = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array} \begin{array}{c} \begin{array}{cccc} b_1 & b_2 & b_3 & b_4 \end{array} \\ \begin{pmatrix} 1 & a & 0 & 1 \\ b & 0 & 1 & 1 \\ 1 & c & d & 1 \end{pmatrix} \end{array}$$

$$\begin{tikzcd}
& & 1 \arrow[dl, dash] \arrow[dr, dash] & \\
& c \arrow[dl, dash] \arrow[dr, dash] & & d \arrow[dl, dash] \\
L = a \arrow[dr, dash] & & b \arrow[dl, dash] & \\
& 0 & &
\end{tikzcd}$$

Definition 2.16. *A fuzzy set can be defined by a matrix with one row where the value denote the membership degree of corresponding object/attribute.*

Example 11. Let A is a fuzzy set on the universe of discourse U where $U = \{\text{Red, Green, Sweet}\}$. According to above definition we can define A as follow.

$A = \{1, 0, 2\}$ where the degree of Red is 1, the degree of Green is 0 and so on.

2.3.1 Basic operations

Let $P, Q : A \to B$, $R : B \to C$, $S : D \to B$ and $T : A \to D$ be L-fuzzy relations. Then we may introduce several operations as follows:

$$\begin{aligned}
\text{Union}, (P \cup Q)(a,b) &:= \{((a,b),(P(a,b) \vee Q(a,b))) \mid (a,b) \in A \times B\} \\
\text{Intersection}, (P \cap Q)(a,b) &:= \{((a,b),(P(a,b) \wedge Q(a,b))) \mid (a,b) \in A \times B\} \\
\text{Implication}, (P \to Q)(a,b) &:= \{((a,b), MAX(l) \mid \underset{l \in L}{\forall}(a \wedge l) \leq b \ and \ (a,b) \in A \times B\} \\
\text{Composition}, (P;R)(a,c) &:= \{((a,c),(\bigvee_{b \in B}(P(a,b) \wedge R(b,c)))) \mid (a,c) \in A \times C\} \\
\text{Left Residue}, (P/S)(a,d) &:= \{((a,d),(\bigwedge_{b \in B}(S(d,b) \to P(a,b)))) \mid (a,d) \in A \times D\} \\
\text{Right Residue}, (Q \backslash T)(b,d) &:= \{((b,d),(\bigwedge_{a \in A}(Q(a,b) \to T(a,d)))) \mid (b,d) \in B \times D\}
\end{aligned}$$

Let us give an example for each of the operations above considering following relations and lattice.

$$P = \begin{array}{c|ccc|} & b_1 & b_2 & b_3 \\ a_1 & 0 & 0.5 & 1 \\ a_2 & 0.7 & 1 & 1 \\ a_3 & 1 & 0 & 0.5 \end{array} \qquad Q = \begin{array}{c|ccc|} & b_1 & b_2 & b_3 \\ a_1 & 0.7 & 0 & 0 \\ a_2 & 1 & 0.5 & 0.7 \\ a_3 & 0.5 & 0.5 & 0.7 \end{array} \qquad R = \begin{array}{c|cccc|} & c_1 & c_2 & c_3 & c_4 \\ b_1 & 1 & 0.5 & 0 & 1 \\ b_2 & 1 & 0.7 & 0.7 & 1 \\ b_3 & 0 & 1 & 0.5 & 0 \end{array}$$

$$S = \begin{array}{c|ccc|} & b_1 & b_2 & b_3 \\ d_1 & 1 & 0 & 0.7 \\ d_2 & 0 & 1 & 1 \\ d_3 & 0.5 & 0.7 & 0.5 \\ d_4 & 0.7 & 0.5 & 0 \end{array} \qquad T = \begin{array}{c|cccc|} & d_1 & d_2 & d_3 & d_4 \\ a_1 & 0.5 & 1 & 0 & 0 \\ a_2 & 0.7 & 0.5 & 1 & 0.7 \\ a_3 & 1 & 0 & 0 & 0.5 \end{array}$$

$$\begin{tikzcd}
1 \arrow[d, dash] \\
0.5 \arrow[d, dash] \\
L = 0.7 \arrow[d, dash] \\
0
\end{tikzcd}$$

We can perform now union($\cup$) and intersection($\cap$) operations using meet ($\wedge$) and join($\vee$) respectively. The join/meet between two elements is simply their maximum/minimum element. The implication between two relation is also performed with the help of the meet operation.

$$P \cup Q = \begin{array}{c c} & \begin{array}{ccc} b_1 & b_2 & b_3 \end{array} \\ \begin{array}{c} a_1 \\ a_2 \\ a_3 \end{array} & \begin{pmatrix} 0.7 & 0.5 & 1 \\ 1 & 1 & 1 \\ 1 & 0.5 & 0.7 \end{pmatrix} \end{array} \qquad P \cap Q = \begin{array}{c c} & \begin{array}{ccc} b_1 & b_2 & b_3 \end{array} \\ \begin{array}{c} a_1 \\ a_2 \\ a_3 \end{array} & \begin{pmatrix} 0 & 0 & 0 \\ 0.7 & 0.5 & 0.7 \\ 0.5 & 0 & 0.5 \end{pmatrix} \end{array}$$

$$P \to Q = \begin{array}{c c} & \begin{array}{ccc} b_1 & b_2 & b_3 \end{array} \\ \begin{array}{c} a_1 \\ a_2 \\ a_3 \end{array} & \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0.5 & 0.7 \\ 0.5 & 1 & 1 \end{pmatrix} \end{array}$$

The composition of L-fuzzy relation is done in similar way as matrix multiplication in linear algebra instead of summing and multiplication we use meet and join respectively.

$$P;R = \begin{array}{c c} & \begin{array}{cccc} c_1 & c_2 & c_3 & c_4 \end{array} \\ \begin{array}{c} a_1 \\ a_2 \\ a_3 \end{array} & \begin{pmatrix} 0.5 & 1 & 0.5 & 0.5 \\ 1 & 1 & 0.7 & 1 \\ 1 & 0.5 & 0.5 & 1 \end{pmatrix} \end{array}$$

The left residue and right residue operation will return following matrices

$$P/S = \begin{array}{c c} & \begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \end{array} \\ \begin{array}{c} a_1 \\ a_2 \\ a_3 \end{array} & \begin{pmatrix} 0 & 0.5 & 0 & 0 \\ 0.7 & 1 & 1 & 1 \\ 0.5 & 0 & 0 & 0 \end{pmatrix} \end{array} \qquad Q\backslash T = \begin{array}{c c} & \begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \end{array} \\ \begin{array}{c} b_1 \\ b_2 \\ b_3 \end{array} & \begin{pmatrix} 0.5 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0.5 \end{pmatrix} \end{array}$$

Definition 2.17. *The symmetric quotient $Q(syq)T : B \to D$ of two fuzzy relations $Q : A \to B$ and $T : A \to D$ is defined by*

$$syq(Q,T)(b,d) = \{((b,d), (\bigwedge_{a \in A} (Q(a,b) \leftrightarrow T(a,d))) \mid (b,d) \in B \times D\}$$

The symmetric quotient between Q and T defined above return the following result

$$syq(Q,T) = \begin{array}{c} \\ b_1 \\ b_2 \\ b_3 \end{array}\begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \\ \hline \end{array}$$

$$syq(Q,T) = \begin{array}{c|cccc} & d_1 & d_2 & d_3 & d_4 \\ b_1 & 0.7 & 0 & 0 & 0 \\ b_2 & 0 & 0 & 0 & 0.7 \\ b_3 & 0 & 0 & 0 & 0.7 \end{array}$$

To convert a fuzzy relation to crisp relation, we need to define two operations called support($\uparrow$) and kernel($\downarrow$). Let consider $P : A \times B$ is a fuzzy relation, then $P^{\uparrow}$ and $P^{\downarrow}$ can be defined as follow

$$P^{\uparrow}(x,y) := \begin{cases} 1 & \text{iff } R(x,y) \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad P^{\downarrow}(x,y) := \begin{cases} 1 & \text{iff } R(x,y) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Example 12. Let P is fuzzy relation as follow:

$$P = \begin{pmatrix} 1 & 0.5 & 0.7 & 0 \\ 1 & 0.5 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0.5 & 0 & 0.7 \end{pmatrix}$$

Then support and kernel operations will return following binary relations respectively

$$P^{\uparrow} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \text{ and } P^{\downarrow} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

2.4 Categories of Relations

In this section, we focus on different types of categories and allegories. We also discuss several constructions within categories or allegories that we need for our framework.

2.4.1 Category Theory

Category theory proposes a tool-set to formalize mathematical concepts and construction. It uses labeled directed graph to explain the structure and theories in mathematics which is called category. The nodes and directed labeled edge of the graph are called object and morphism respectively.

Definition 2.18. Category $\mathcal{C}$ consists of

1. *a class of objects $Obj_{\mathcal{C}}$,*

2. *a class of morphisms, maps or arrows $\mathcal{C}[A,B]$ for every pair of objects A and B where each morphism has source and target. A morphism f from a to b can be stated as $f : a \to b$,*

3. *an associative binary operation ; called composition of morphism mapping each pair of morphisms f in $\mathcal{C}[A,B]$ and g in $\mathcal{C}[B,C]$ to a morphism $f;g$ in $\mathcal{C}[A,C]$,*

4. *for every object A a morphism, $\mathbb{I}_A$ such that for all f in $\mathcal{C}[A,B]$ and g in $\mathcal{C}[C,A]$ we have $\mathbb{I}_A \ ; \ f = f$ and $g \ ; \ \mathbb{I}_A = g$.*

If f is a morphism in $\mathcal{C}[A,B]$, we will denote it by $f : A \to B$.

We can use diagrams to illustrate different relations in categories where node and directed arrow represent object and morphism in category respectively. A path of length two in the graph express the composition between the morphism denoted by the edges in the path. For example, consider three morphism $f : A \to B$ and $g : B \to C$ and $h : A \times C$ in category $\mathcal{C}$. Figure 5.3 represents the relation $f;g = h$. i.e. the composition between f and g is equal to h.

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "h"'] & B \arrow[d, "g"] \\
 & C
\end{tikzcd}$$

Figure 2.2: Visual representation of categories.

2.4.2 Allegories

An allegory is an abstraction of the category **Rel** of sets and the morphisms are considered to be binary relations between them. The following is a general definition of allegories as defined in [14].

Definition 2.19. *An allegory R is a category that satisfies the following.*

1. *For all objects A and B, the class of morphisms $R[A,B]$ is a lower semilattice. The meet and the induced ordering are denoted by $\sqcap, \sqsubseteq$ respectively. The elements in $R[A,B]$ are called relations.*

2. *There is a monotone operation $\smile$ (converse operation) such that for all relations $Q : A \to B$ and $S : B \to C$ the following is true:*

$$(Q;S)^{\smile} = S^{\smile};Q^{\smile} \text{ and } (Q^{\smile})^{\smile} = Q.$$

3. *For all relations $Q : A \to B$ and $R, S : B \to C$ the following is true:*

$$Q;(R \sqcap S) \sqsubseteq Q;R \sqcap Q;S$$

4. *For all relations $Q : A \to B, R : B \to C$ and $S : A \to C$ the following is true (modular law):*

$$Q;R \sqcap S \sqsubseteq Q;(R \sqcap Q^{\smile};S)$$

Lemma 2.1. *Let R be an allegory, A,B,C be objects of R and $Q, R : A \to B$, $S : B \to C$,$T : A \to C$, and $U, V : A \to A$. Then we have*

1. $\mathbb{I}_A = \mathbb{I}_A$,
2. $(Q \sqcap R);S \sqsubseteq Q;S \sqcap R;S$,
3. *For both argument ; is monotone,*
4. $Q;S \sqcap T \sqsubseteq (Q \sqcap T;S^{\smile});S$,
5. $Q;S \sqcap T \sqsubseteq (Q \sqcap T;S^{\smile});(S \sqcap Q^{\smile};T)$,
6. $Q \sqsubseteq Q;Q^{\smile};Q$,
7. $\mathbb{I} \sqcap (U \sqcap V);(U \sqcap V)^{\smile} = \mathbb{I} \sqcap U;V^{\smile} = \mathbb{I}_A \sqcap V;U^{\smile}$,
8. $Q = (\mathbb{I}_A \sqcap Q;Q^{\smile});Q = Q;(\mathbb{I}_B \sqcap Q^{\smile};Q)$.

Definition 2.20. *Let R be an allegory and relation $Q : A \to B$. Then we call*

1. *Q univalent iff $Q^{\smile};Q \sqsubseteq \mathbb{I}_B$,*
2. *Q total iff $\mathbb{I}_A \sqsubseteq Q;Q^{\smile}$,*

3. *Q a map iff Q is univalent and total,*

4. *Q injective iff $Q^{\smile}$ is univalent,*

5. *Q surjective iff $Q^{\smile}$ is total,*

6. *Q bijective iff $Q^{\smile}$ is a map,*

7. *Q a bijective iff Q is a bijective map.*

2.4.3 Distributive Allegories

Every $R[A,B]$ in a distributive allegory is a distributive lattice with a least element. The formal definition is:

Definition 2.21. *A distributive allegory R is an allegory satisfying following properties for all relations $Q : A \to B$ and $R, S : B \to C$*

1. *Every $R[A,B]$ is a distributive lattice with a least element where union and the least element are denoted by $\sqcup$ and $\perp\!\!\!\perp_{AB}$ respectively,*

2. *$Q;\perp\!\!\!\perp_{BC}=\perp\!\!\!\perp_{AC}$,*

3. *$Q;(R \sqcup S) = Q;R \sqcup Q;S$.*

By adding the residual operation of a relation algebra to a distributive structure, we can get a division allegory as defined in [14].

Lemma 2.2. *Let R be a distributive allegory. If $Q, R : A \to B$ and $S : B \to C$, we have,*

1. *$\perp\!\!\!\perp^{\smile}_{AB}=\perp\!\!\!\perp_{BA}$,*

2. *$\perp\!\!\!\perp_{CA};Q =\perp\!\!\!\perp_{CB}$,*

3. *$(Q \sqcup R)^{\smile} = Q^{\smile} \sqcup R^{\smile}$,*

4. *$(Q \sqcup R);S = Q;S \sqcap R;S$.*

2.4.4 Division Allegories

According to the hierarchy of allegories, the next step after distributive allegories are division allegories. In the division allegories ; is a lower adjoint.

Definition 2.22. *A distributive allegory $\mathcal{R}$ is called division allegory iff for all relations $R : B \to C$ and $S : A \to C$ there is a left residual $S/R : A \to B$ such that for all $Q : A \to B$ the following holds*

$$Q;R \sqsubseteq S \Leftrightarrow Q \sqsubseteq S/R.$$

An upper right adjoint is also exists for ; in division allegory which is called right residual. For relations $Q : A \to B$ and $S : A \to C$, right residual is defined as follow

$$Q\backslash S = (S^{\smile}/Q^{\smile})^{\smile}$$

The symmetric version of the residuals defined as

$$syq(Q,R) = (Q\backslash R) \sqcap (Q^{\smile}/R^{\smile})$$

The next lemma provides some basic properties of the symmetric quotient, and all the proofs can be found in [14].

The following lemma shows some fundamental properties of symmetric quotients [16].

Lemma 2.3. *Let $\mathcal{R}$ be a division allegory. If $Q : A \to B, R : A \to C, S : A \to D$ are arbitrary relations and $f : D \to B$ is a mapping. Then we have*

1. $f; syq(Q,R) = syq(Q;f^{\smile}, R)$,
2. $syq(Q,R)^{\smile} = syq(R,Q)$,
3. $syq(Q,R); syq(R,S) \sqsubseteq syq(Q,S)$.

Lemma 2.4. *Let $\mathcal{R}$ be a division allegory. If $Q, Q_1, Q_2 : A \to B,\ R, R_1, R_2 : B \to C$ and $S, S_1, S_2 : A \to C$, then we have,*

1. $Q \sqsubseteq (Q;R)/R$ *and* $R \sqsubseteq Q\backslash(Q;R)$,
2. $(Q\backslash R);(R\backslash S) \subseteq (Q\backslash S)$,
3. $(S/R);R \sqsubseteq S$ *and* $Q;(Q\backslash S) \sqsubseteq S$,
4. $S/(Q\backslash S) \sqsubseteq Q$ *and* $(S/R)\backslash S \sqsubseteq R$,

5. *$Q_2 \sqsubseteq Q_1$, $R_2 \sqsubseteq R_1$ and $S_2 \sqsubseteq S_1$ implies $S_1/R_1 \sqsubseteq S_2/R_2$ and $Q_1\backslash S_1 \sqsubseteq Q_2\backslash S_2$,*
6. *$(S_1 \sqcap S_2)/R = (S_1/R) \sqcap (S_2/R)$ and $Q\backslash(S_1 \sqcap S_2) = (Q\backslash S_1) \sqcap (Q\backslash S_2)$,*
7. *$S/(R_1 \sqcup R_2) = (S/R_1) \sqcup (S/R_2)$ and $(Q_1 \sqcup Q_2)\backslash S = (Q_1\backslash S) \sqcup (Q_2\backslash S)$.*

Lemma 2.5. *Let R be a division allegory. If $Q : A \to B, R : B \to C, S : A \to C, F : D \to A$, and $G : C \to E$ then we have,*

1. *$S/\mathbb{I}_C = S$ and $\mathbb{I}_A\backslash S = S$,*
2. *$F;(S/R) \sqsubseteq (F;S)/R$ and $(Q\backslash S);G \sqsubseteq Q\backslash(S;G)$,*
3. *If F and $G^{\smile}$ are mappings, then in both properties of (2) equality holds,*
4. *$S/R \sqsubseteq (S;G)/(R;G)$ and $Q\backslash S \sqsubseteq (F;Q)\backslash(F;S)$,*
5. *If G and $F^{\smile}$ are total and injective, then in both properties of (4) equality holds.*

2.4.5 Dedekind Category

Definition 2.23. *A division allegory R is called Dedekind category iff every $R[A,B]$ is a complete Brouwerian lattice. Meet, join, the inducing ordering, the least and the greatest element are denoted by $\sqcap, \sqcup, \sqsubseteq, \perp\!\!\!\perp_{AB}$, and $\top\!\!\!\top_{AB}$ respectively. Dedekind category satisfies the following properties*

1. There is a converse operation $\smile$ (monotone operation) changing a relation $Q : A \to B$ to $Q^{\smile} : B \to A$ such that the following are true

$$(Q;R)^{\smile} = R^{\smile};Q^{\smile},$$
$$(Q^{\smile})^{\smile} = Q.$$

2. Satisfies the following modular law

$$(Q;R) \sqcap S \sqsubseteq Q;(R \sqcap (Q^{\smile};S)).$$

3. For all $S : A \to C$ and $R : B \to C$, there is a relation $S/R : A \to B$ which is called left residual such that for all $X : A \to B$ the following is true

$$X;R \sqsubseteq S \Leftrightarrow X \sqsubseteq S/R.$$

4. The right residual $R\backslash S$ can be formulated by $R\backslash S = (S/R)^{\smile}$ and it is identified as

$$R;X \sqsubseteq S \Leftrightarrow X \sqsubseteq S/R.$$

Lemma 2.6. *Let $\mathcal{R}$ be a Dedekind category. Then for all objects A and B in $\mathcal{R}$ the following are true.*

1. $\mathbb{T}^{\smile}_{AB}$=$\mathbb{T}_{AB}$,
2. $\mathbb{T}_{AA};\mathbb{T}_{AB}$=$\mathbb{T}_{AB};\mathbb{T}_{BB}$=$\mathbb{T}_{AB}$,
3. $\mathbb{T}_{AB}$=$\mathbb{T}_{AB};\mathbb{T}_{BA};\mathbb{T}_{AB}$.

Lemma 2.7. *Let $\mathcal{R}$ be a Dedekind category, $Q : A \to B, R : B \to C, S : A \to D$ and$T : D \to C$. Then the following are true*

1. $(Q \sqcap S;\mathbb{T}_{DB});R = Q;R \sqcap S;\mathbb{T}_{DC}$,
2. $\mathbb{I}_A \sqcap Q;Q^{\smile} = \mathbb{I}_A \sqcap Q;\mathbb{T}_{BA}$= $\mathbb{I}_A$⊓ $\mathbb{T}_{AB};Q^{\smile}$.

Other important properties of Dedekind categories are related to partial identities, and they are shown in next lemma [16].

Lemma 2.8. *Let $\mathcal{R}$ be a Dedekind category, $S : A \to A$ is a partial identities, and $R : C \to A$, $U : A \to B$. Then the following are true*

1. $S = \mathbb{I}_A \sqcap S;\mathbb{T}_{AA}$= $\mathbb{I}_A$ ⊓ $\mathbb{T}_{AA};S$,
2. $R;S = R \sqcap \mathbb{T}_{CA};S$ and $S;U = U \sqcap S;\mathbb{T}_{AB}$,
3. $S^{\smile} = S$.

The proof of the above lemma is an easy exercise, and can be found in [14, 16]. Since L–Rel$[A,B]$ is a complete Heyting algebra, L–Rel forms a Dedekind category. However, the language of Dedekind categories is not rich enough to define crispness as property of relations.

Therefore, we need to define a new category that covers the crispness property of L–fuzzy relations. Adding two arrow operations, the up-arrow (↑) and the down-arrow (↓), gives us a new structure called the arrow category [15].

2.4.6 Arrow Category

Definition 2.24. *An arrow category $\mathcal{A}$ is an extension of Dedekind category with* $\perp\!\!\!\perp_{AB} \neq \top\!\!\!\top_{AB}$ *for all objects A and B together with operations $\uparrow, \downarrow$ satisfying the following properties for all $Q, R : A \to B, S : B \to A$ and $T : B \to A$*

1. $R^{\uparrow}, R^{\downarrow} : A \to B$,

2. $(\uparrow, \downarrow)$ is a Galois correspondence, i.e., $Q^{\uparrow} \sqsubseteq R$ iff $Q \sqsubseteq R^{\downarrow}$,

3. $(S^{\smile}; T^{\downarrow})^{\uparrow} = S^{\uparrow\smile}; T^{\downarrow}$,

4. $(Q \sqcap R^{\downarrow})^{\uparrow} = Q^{\uparrow} \sqcap R^{\downarrow}$.

Now we can show the algebraic theory of L-fuzzy relations using the next lemma as defined in [14].

Lemma 2.9. *Let L be a complete Heyting algebra with $0 \neq 1$. Then L–Rel together with $\uparrow$ and $\downarrow$ is an arrow category.*

The following lemma summarizes some basic properties of arrow categories as shown in [14].

Lemma 2.10. *Let $\mathcal{A}$ be an arrow category and $Q, R : A \to B, S : B \to C, T : A \to C$. Then the following are true:*

1. $\mathbb{I}_A^{\uparrow} = \mathbb{I}_A \neq \perp\!\!\!\perp_{AA}$,

2. $R^{\downarrow\uparrow} = R^{\downarrow}$,

3. $R^{\downarrow\uparrow} = R^{\uparrow}$,

4. $\uparrow, \downarrow$ are closure and kernel operations respectively,

5. $R = R^{\uparrow}$ iff $R^{\downarrow} = R^{\uparrow}$ iff $R^{\downarrow} = R$,

6. $\perp\!\!\!\perp_{AB}^{\uparrow} = \perp\!\!\!\perp_{AB}$ and $\top\!\!\!\top_{AB}^{\top\!\!\!\top} = \top\!\!\!\top_{AB}$,

7. $(R^{\smile}; S^{\uparrow})^{\uparrow} = R^{\uparrow\smile}; S^{\uparrow}$,

8. $R^{\smile\uparrow} = R^{\uparrow\smile}$ and $R^{\smile\downarrow} = R^{\downarrow\smile}$,

9. $(R; S^{\downarrow})^{\uparrow} = R^{\uparrow}; S^{\downarrow}$ and $(R^{\downarrow}; S)^{\uparrow} = R^{\downarrow}; S^{\uparrow}$,

10. $(R; S^{\uparrow})^{\uparrow} = R^{\uparrow}; S^{\uparrow}$ and $(R^{\uparrow}; S)^{\uparrow} = R^{\uparrow}; S^{\uparrow}$,

11. $(Q \sqcap R^{\uparrow})^{\uparrow} = Q^{\uparrow} \sqcap R^{\downarrow}$,

12. For all nonzero ideal relations $\jmath^{\uparrow} = \mathbb{T}_{AB}$ holds,

13. $R^{\downarrow} \to Q^{\uparrow} = (R \to Q^{\uparrow})^{\downarrow} \sqsubseteq (R \to Q)^{\downarrow}$ and $(R \to Q)^{\uparrow} \sqsubseteq R^{\uparrow} \to Q^{\downarrow}$,

14. $Q^{\uparrow} \backslash T^{\downarrow} = (Q^{\uparrow} \backslash T)^{\downarrow} \sqsubseteq (Q \backslash T)^{\downarrow}$ and $(Q \backslash T)^{\uparrow} \sqsubseteq Q^{\downarrow} \backslash T^{\uparrow}$,

15. $T^{\downarrow}/S^{\uparrow} = (T/S^{\uparrow})^{\downarrow} \sqsubseteq (T/S)^{\downarrow}$ and $(T/S)^{\uparrow} \sqsubseteq T^{\uparrow}/S^{\downarrow}$.

The full proof of above lemma can be found in [14]. The next lemma shows a collection of closure properties of the class of crisp relations, and the proof can be found in [14].

Lemma 2.11. *Let $\mathcal{A}$ be an arrow category and $Q_i, Q, T : A \to B$ for $i \in I, R : A \to C$, and $S : B \to C$ crisp relations. Then the following are true:*

1. $\bigsqcup_{i \in I} Q_i$ and $\bigsqcap_{i \in I} Q_i$ are crisp,
2. $Q^{\smile}$ is crisp,
3. $Q; S$ is crisp,
4. R/S and $Q \backslash R$ are crisp,
5. $Q \to T$ is crisp.

There are some other properties of an arrow categories and they are collected in the following lemmas.

Lemma 2.12. *Let $\mathcal{A}$ be an arrow category and $Q : A \to B$ and $R : B \to C$ crisp relations. Then it satisfies the following*

$$Q^{\downarrow}; R^{\downarrow} \sqsubseteq (Q; R)^{\downarrow}.$$

Lemma 2.13. *Let $\mathcal{A}$ be an arrow category and $Q : A \to B$ and $R : C \to B$ crisp relations. $f : A \to C$ is a crisp map. Then the following holds:*

1. $f; Q^{\downarrow} = (f; Q)^{\downarrow}$,
2. $f; syq(Q, R)^{\downarrow} = syq(Q; f^{\downarrow}, R)^{\downarrow}$,
3. $f; (Q \backslash R)^{\downarrow} = (Q; f^{\smile} \backslash R)^{\downarrow}$,
4. $(Q \backslash R)^{\downarrow}; f^{\smile} = (Q \backslash R; f^{\smile})^{\downarrow}$.

The next lemma shows another important property of symmetric quotients in arrow categories.

Lemma 2.14. *Let A be an arrow category and $Q : A \to B$ and $R : C \to B$ crisp relations. If $syq(Q,R)^{\downarrow}$ is surjective, then the following holds:*

$$Q; syq(Q,R)^{\downarrow} = R.$$

2.4.7 Relational constructions in Dadekind categories

This section describes about relational products and splitting which is a type of relational construction in Dedekind categories [14].

Definition 2.25. *Let A and B be objects of a Dedekind category. An object $A \times B$, together with two relations $\pi : A \times B \to A$ and $\rho : A \times B \to B$ s called a relational product of A and B iff the following holds*

1. $\pi^{\smile}; \pi \subseteq \mathbb{I}_A$,
2. $\rho^{\smile}; \rho \subseteq \mathbb{I}_B$,
3. $\pi^{\smile}; \rho = \mathbb{T}_{AB}$,
4. $\pi; \pi^{\smile} \cap \rho; \rho^{\smile} = \mathbb{I}_{A\times B}$.

R has relational products iff for every pair of objects a relational product does exist.

Now if any direct products is given by the following projections $\pi = A \times B \to A$ and $\rho = A \times B \to B$, we can define a new operator as an operation for relations [10]. Let P be relation $D \to A$ and Q be relation $D \to B$, then the fork operator is

$$P \otimes Q = P; \pi^{\smile} \cap Q; \rho^{\smile}$$

The fork operator is illustrated in Figure 2.3.

$$\begin{tikzcd}
 & D \arrow[dl, "P"'] \arrow[d, "P\otimes Q"] \arrow[dr, "Q"] & \\
A \arrow[r] & A \times B & B \arrow[l]
\end{tikzcd}$$

Figure 2.3: A visual representation of the fork operator

Definition 2.26. *Let $Q : A \to A$ be a symmetric idempotent relation, i.e., $Q^\smile = Q$ and $Q;Q = Q$. An object B together with a relation $R : B \to A$ is called a splitting of Q (or R splits Q) iff $R;R^\smile = \mathbb{I}_B$ and $R^\smile;R = Q$.*

Splitting is unique up to isomorphism. If Q is a partial identity, the object B of the splitting corresponds to the subset given by Q. Analogously, if Q is an equivalence relation, B corresponds to the set of equivalence classes.

2.4.8 Fuzzy power

In this section, we going to describe the fuzzy power sets [17].

Definition 2.27. *An object $P(A)$ together with a relation $\varepsilon : A \to P(A)$ is called a relational power iff the following holds*

1. $syq(\varepsilon, \varepsilon) = \mathbb{I}_{P(A)}$,
2. $syq(R, \varepsilon)$ is total for every $R : A \to B$.

The relational power is an abstract version of the power set of a set. In particular, it emphasizes extensionality ($syq(\varepsilon, \varepsilon) = \mathbb{I}$) as a basic property of sets.

Normally, in a fuzzy context we are usually interested in the L-power set L^A of a set, i.e., the set of (L-characteristic) functions from A to L. We will now look at the definition of a relational power in terms of the set L.

Definition 2.28. *An object $P_L(A)$ together with a relation $\varepsilon : A \times P_L(A)$ is called a fuzzy relational power iff following holds*

1. $syq(\varepsilon, \varepsilon)^\downarrow = \mathbb{I}_{P_L(A)}$,
2. $syq(R, \varepsilon)^\downarrow$ is total for every $R : A \to B$.

Definition 2.29. *The right residual relation $e = \varepsilon\backslash\varepsilon : L^X \to L^X$ relates two fuzzy subsets A and B of X by $e(A,\ B) = \bigwedge_{x\in X} \varepsilon(x, A) \to \varepsilon(x, B)$*

Please note that in the classical case, i.e., L=2, we have $e(A, B)$ iff $A \subseteq B$.

Chapter 3

Formal Concept Analysis

3.1 Formal Concept Analysis

In this section, we are going to discuss about background theory of formal concept analysis. We proposed relational algebraic formulas later to explain the theory of formal concept analysis and it is one of our main contribution in the thesis. The basic notions of formal concept analysis are those of a formal context and a formal concept [8].

Definition 3.1. *A formal context $\mathbb{K} := (G, M, I)$ is a tabular structure consisting of two sets G and M and a binary relation I (also called Incidence relation) between the two sets. The elements of G are called the objects and the elements of M are called the attributes of the context.*

In order to express that an object g is in a relation I with an attribute m, we write gIm or $(g, m) \in I$ and read it as "the object g has the attribute m".

Table 3.1: Formal Context

	Red	Green	Yellow	Sweet	Sour
Apple	1	0	0	1	1
Banana	0	0	1	1	0
Cherry	1	0	0	1	0
Lemon	0	0	1	0	1

Definition 3.2. *The operator Δ determines the set of attributes common to the every objects in $A \subseteq G$. Correspondingly, for a set $B \subseteq M$, the operator ∇ determines the the set of objects which have all attributes in B.*

Please note that Δ and ∇ corresponds to the sets of lower and upper bounds in the case of ordered set [3].

$$A^{\Delta} = \{m \in M \mid \forall g \in A : gIm\}$$

$$B^{\nabla} = \{g \in G \mid \forall m \in B : gIm\}$$

Example 13. For example, let $A = \{\text{Apple}\}$ and $B = \{\text{Apple, Cherry}\}$ two object subsets from Table 3.1. Now if we apply the derivation operator Δ on these object sets, we will get output as follow.

$$A^{\Delta} = \{\text{Red, Sweet, Sour}\}$$
$$B^{\Delta} = \{\text{Red, Sweet}\}$$

Let consider two attribute subsets $C = \{\text{Sour}\}$ and $D = \{\text{Yellow}, \text{Sweet}\}$ from Table 3.1. The derivation operator ∇ will produces following output

$$C^{\nabla} = \{\text{Apple, Lemon}\}$$
$$D^{\nabla} = \{\text{Banana}\}$$

Definition 3.3. *A formal concept of the context (G, M, I) is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A^{\Delta} = B$ and $B^{\nabla} = A$. The sets A and B are called the extent and intent of the concept respectively. $\mathfrak{B}(G, M, I)$ denotes the set of all concepts of the context (G, M, I) which is defined as follow*

$$\mathfrak{B}(G, M, I) = \{(A, B) \in 2^G \times 2^M \mid A = B^{\nabla} \wedge B = A^{\Delta}\}$$

Example 14. Let $A = \{\text{Apple}, \text{Cherry}\} \subseteq G$ and $B = \{\text{Red}, \text{Sweet}\} \subseteq M$.

$$A^{\Delta} = \{\text{Red}, \text{Sweet}\} \text{ and}$$
$$B^{\nabla} = \{\text{Apple}, \text{Cherry}\}$$

Since $A = B^{\nabla}$ and $B = A^{\Delta}$, so we can conclude $(\{\text{Apple}, \text{Cherry}\}, \{\text{Red}, \text{Sweet}\})$ is a concept. On a contrary, $(\{\text{Cherry}\}, \{\text{Red}, \text{Sweet}\})$ is not a concept.

Definition 3.4. *If (A_1, B_1) and (A_2, B_2) are concepts of a context, then (A_1, B_1) is called a subconcept of (A_2, B_2), provided that $A_1 \subseteq A_2$ (which is equivalent to $B_2 \subseteq B_1$) [8].*

We can denote this hierarchical relationship as $(A_1, B_1) \leq (A_2, B_2)$ where (A_2, B_2) is called the super-concept of (A_1, B_1). The relation $\leq$ is called the hierarchical order (or simply order) of the concepts. The set of all concepts of (G, M, I) ordered in this way is denoted by $\underline{\mathfrak{B}}(G, M, I)$ and is called the concept lattice of the context (G, M, I) which can be illustrated using a diagram called hasse diagram.

Example 15. For example, consider two concepts as follow

$$(A_1, B_1) = (\{\text{Apple}, \text{Cherry}\}, \{\text{Red}, \text{Sweet}\})$$
$$(A_2, B_2) = (\{\text{Apple}, \text{Banana}, \text{Cherry}\}, \{\text{Sweet}\})$$

Here, (A_1, B_1) is a subconcept of (A_2, B_2) satisfying the condition $\{\text{Apple}, \text{Cherry}\} \subseteq \{\text{Apple}, \text{Banana}, \text{Cherry}\}$. Their hierarchy is denoted by $(A_1, B_1) \leq (A_2, B_2)$.

Definition 3.5. *If (A_1, B_1) and (A_2, B_2) are two concepts of a context, then the greatest common subconcept or infimum (least general superconcept or supremum) can be defined using meet (join) as follow.*

$$(A_1, B_1) \wedge (A_2, B_2) = (A_1 \cap A_2, (B_1 \cup B_2)^{\nabla\Delta})$$
$$(A_1, B_1) \vee (A_2, B_2) = ((A_1 \cup A_2)^{\Delta\nabla}, (B_1 \cap B_2))$$

Example 16. Let consider two concepts as follow

$$(A_1, B_1) = (\{\text{Apple}, \text{Cherry}\}, \{\text{Red}, \text{Sweet}\})$$
$$(A_2, B_2) = (\{\text{Apple}, \text{Banana}, \text{Cherry}\}, \{\text{Sweet}\})$$

The infimum and supremum of these two concepts is calculated below according to the definition above

$$(A_1 \cap A_2) = \{\text{Apple}, \text{Cheery}\},\ (B_1 \cup B_2)^{\nabla\Delta} = \{Red, Sweet\}$$
$$(A_1 \cup A_2)^{\Delta\nabla} = \{\text{Apple}, \text{Cheery}, \text{Banana}\},\ (B_1 \cap B_2) = \{\text{Sweet}\}$$
$$(A_1, B_1) \wedge (A_2, B_2) = (\{\text{Apple}, \text{Cheery}\}, \{\text{Red}, \text{Sweet}\})$$
$$(A_1, B_1) \vee (A_2, B_2) = (\{\text{Apple}, \text{Cheery}, \text{Banana}\}, \{\text{Sweet}\})$$

3.2 Fuzzy Formal Concept Analysis

Formal concept analysis can handle uncertain and vague information in the context. There are lots of previous research works which exploit the fuzziness into FCA and extend FCA model to fuzzy formal contexts. It is mainly the extension of the original formal concept analysis by setting truth degree for the propositions "object g has attribute m" in fuzzy formal contexts. Truth degree are taken from an appropriate scale L[0,1] of truth degrees. Thus, the entries of a formal context representing objects and attributes become degrees from L instead of Boolean values as is the case of the basic setting of FCA. This extension is known as Fuzzy Formal Concept Analysis(FFCA). In this section, we are going to explain some definitions from Fuzzy Formal Concept Analysis (FFCA) as our proposed algebraic framework can handle Boolean as well as fuzzy settings.

Definition 3.6. A fuzzy formal context is an ordered triple $\mathbb{K}_f := (G, M, I_f)$ in which G and M are non-empty (classical) sets, and I_f fuzzy binary relation.

Let G and M be sets of objects and attributes, respectively, I_f be a fuzzy relation between G and M. That is, $I_f : G \times M \to L$ assigns to each $g \in G$ and each $m \in M$ a truth degree $I_f(g, m) \in L$ to which object g has attribute m.

Definition 3.7. We can define two operator Δ and ∇ respectively for a fuzzy set $A \in L^G$ of objects and fuzzy set $B \in L^M$ of attributes as follow

$$A^{\Delta}(m) = \bigwedge_{g \in G} (A(g) \to I_f(g, m))$$
$$B^{\nabla}(g) = \bigwedge_{m \in M} (B(m) \to I_f(g, m))$$

Table 3.2: Fuzzy Formal Context

	Red	Green	Yellow	Sweet	Sour
Apple	1	2	0	1	2
Banana	0	2	1	1	2
Cherry	1	0	0	1	2
Lemon	0	2	1	0	1

The fuzzy formal context represented in the Table 3.2 is based on a three element lattice L in Figure 3.1

$$\begin{tikzcd} 1 \arrow[d, dash] \\ L = 2 \arrow[d, dash] \\ 0 \end{tikzcd}$$

Figure 3.1: Three element lattice

Example 17. Let $A = \{2, 0, 2, 0\} \subseteq G$ and $B = \{0, 0, 1, 0, 1\} \subseteq M$ are two fuzzy subsets from Table 3.2.

$$A^{\Delta} = \{0, 0, 0, 1, 2\}$$
$$B^{\nabla} = \{0, 2, 0, 1\}$$

Lets explain the detail calculation according to the Definition 3.7

$A^{\Delta}(\text{Red}) = A(\text{Apple}) \to I_f(\text{Apple, Red}) \wedge A(\text{Banana}) \to I_f(\text{Banana, Red}) \wedge A(\text{Cherry}) \to$

$I_f(\text{Cherry, Red}) \;\wedge\; A(\text{Lemon}) \to I_f(\text{Lemon, Red})$
$= (2 \to 1) \;\wedge\; (1 \to 0) \;\wedge\; (2 \to 1) \;\wedge\; (0 \to 0)$
$= 1 \wedge 0 \wedge 1 \wedge 1$
$= 0$

$A^{\Delta}(\text{Sour}) = A(\text{Apple}) \to I_f(\text{Apple, Sour}) \;\wedge\; A(\text{Banana}) \to I_f(\text{Banana, Sour}) \;\wedge\; A(\text{Cherry}) \to I_f(\text{Cherry, Sour}) \;\wedge\; A(\text{Lemon}) \to I_f(\text{Lemon, Sour})$
$= (2 \to 2) \;\wedge\; (1 \to 2) \;\wedge\; (2 \to 2) \;\wedge\; (0 \to 1)$
$= 1 \wedge 2 \wedge 1 \wedge 1$
$= 2$

We can follow the same process as above for other attributes. Similiarly, we can determine the degree of each object for the attribute set B.

Definition 3.8. *Let (G, M, I_f) be a fuzzy context. A pair $(A, B) \in L^G \times L^M$ is called a formal concept if $A^{\nabla} = B$ and $B^{\Delta} = A$. The fuzzy sets A and B are called the extent and intent of the concept respectively.*
The set of all formal fuzzy concepts can be defined as follow

$$\mathfrak{B}(G, M, I_f) = \{(A, B) \in L^G \times L^M \mid A^{\Delta} = B \wedge B^{\nabla} = A\}$$

Chapter 4

Relational Concept Analysis

4.1 Concept Lattice in Allegories

We will now formulate concept analysis in the language of relations. The advantage is that same formulas can be applied in the classical and fuzzy situation. Often we will explain the formulas just in the classical case. Our first goal is to show that it is sufficient to consider sets of objects (resp. sets of attributes) instead of pairs of those when dealing with formal concepts. In order to do so, we start with the the functions ∇ and Δ mapping sets of objects to sets of attributes and vice versa.

Definition 4.1. Let $R : G \to M$ be a relation and $\varepsilon : G \to 2^G(M \to 2^M)$ be the is-element-relation. Then we can define operators Δ and ∇ as follow

$$\Delta = syq(R^\smile/\varepsilon^\smile, \varepsilon)^\downarrow$$
$$\nabla = syq(R/\varepsilon^\smile, \varepsilon)^\downarrow$$

The next lemma shows that the composition if the two function from the previous definition can be written as a simple construction between the power objects. The first composition operator $\Delta;\nabla$ generates a relation between all object subsets from the context which is true for the object subset and their corresponding closure set. The other composition operator $\nabla;\Delta$ do the same operation on the attribute subset from the context.

Lemma 4.1. Suppose $R : G \to M$ be a relation and $\varepsilon : G \to 2^G(M \to 2^M)$ be the is-element-relation. Then we have

1. $\Delta;\nabla = syq(R/(\varepsilon\backslash R), \varepsilon)^\downarrow$
2. $\nabla;\Delta = syq(R^\smile/(\varepsilon\backslash R^\smile), \varepsilon)^\downarrow$

Proof. 1) We immediately compute

$$\begin{aligned}
\Delta;\nabla &= syq(R^{\smile}/\varepsilon^{\smile},\varepsilon)^{\downarrow};syq(R/\varepsilon^{\smile},\varepsilon)^{\downarrow} \\
&= syq((R/\varepsilon^{\smile});syq(\varepsilon,R^{\smile}/\varepsilon^{\smile})^{\downarrow},\varepsilon)^{\downarrow} && \text{Lemma 2.3(1-2)} \\
&= sqy(R/syq(R^{\smile}/\varepsilon^{\smile},\varepsilon)^{\downarrow};\varepsilon^{\smile},\varepsilon)^{\downarrow} && \text{Lemma 2.5(3)} \\
&= syq(R/(R^{\smile}/\varepsilon^{\smile})^{\smile},\varepsilon)^{\downarrow} \\
&= syq(R/(\varepsilon\backslash R),\varepsilon)^{\downarrow} \\
&= syq(R^{\smile}/(\varepsilon\backslash R^{\smile}),\varepsilon)^{\downarrow}.
\end{aligned}$$

2) is shown analogously.

□

Now we are going to define the partial identity on $L^G \times L^M$ of formal concepts. In addition we define the corresponding partial identities on objects resp. attributes only. We used the derivation and composition operators defined above to produce concepts from Boolean context as well as multi-valued context.

Definition 4.2. Let $R : G \to M$ be a relation and $\varepsilon : G \to L^G(M \to L^M)$ be the is-element-relation. Then we define the partial identities $i_F : L^G \times L^M \to L^G \times L^M$, $i_G : L^G \to L^G$, $i_M : L^M \to L^M$ by

$$\begin{aligned}
i_F &= \pi;\Delta;\rho^{\smile} \ \cap\ \rho;\nabla;\pi^{\smile} \ \cap\ \mathbb{I}, \\
i_G &= \Delta;\nabla \cap\ \mathbb{I}, \\
i_M &= \nabla;\Delta \cap\ \mathbb{I}.
\end{aligned}$$

All three partial identities defined above are crisp relations and we have:

$$\begin{aligned}
i_F(A,B) &\leftrightarrow A^{\Delta} = B \wedge B^{\nabla} = A \\
i_G(A) &\leftrightarrow A^{\Delta\nabla} = A \\
i_M(B) &\leftrightarrow B^{\nabla\Delta} = B
\end{aligned}$$

We already known that a concept (A_1, B_1) is sub-concept of (A_2, B_2) iff $A_1 \subseteq A_2$ or equivalently $B_2 \subseteq B_1$. Now we going to prove that the two properties $A_1 \subseteq A_2$ and $B_2 \subseteq B_1$ for two concepts (A_1, B_1) and (A_2, B_2) are equivalent using following lemma.

Lemma 4.2. Let $R : G \to M$ be a relation. For $A \subseteq G$ and $B \subseteq M$, $A^{\Delta} = B$ iff $B^{\nabla} = A$. We can define this lemma algebraically as follow

$$i_F;\pi;e;\pi^{\smile};i_F = i_F;\rho;e^{\smile};\rho^{\smile};i_F$$

Proof. First, we have

$$\begin{aligned}
e;\Delta;\varepsilon^{\smile} &= (\varepsilon\backslash\varepsilon);syq(R^{\smile}/\varepsilon^{\smile},\varepsilon)^{\downarrow};\varepsilon^{\smile} \\
&= (\varepsilon\backslash\varepsilon);(R^{\smile}/\varepsilon^{\smile})^{\smile} && \text{Lemma 2.14 and 2.3(2)} \\
&= (\varepsilon\backslash\varepsilon);(\varepsilon\backslash R) \\
&\subseteq \varepsilon\backslash R && \text{Lemma 2.4(2)} \\
&= (R^{\smile}/\varepsilon^{\smile})^{\smile} \\
&= syq(R^{\smile}/\varepsilon^{\smile},\varepsilon)^{\downarrow};\varepsilon^{\smile} && \text{Lemma 2.14 and 2.3(2)} \\
&= \Delta;\varepsilon^{\smile}
\end{aligned}$$

So that $\Delta^{\smile};e;\Delta;\varepsilon^{\smile} \subseteq \Delta^{\smile};\Delta;\varepsilon^{\smile} \subseteq \varepsilon^{\smile}$ follows since Δ is univalent. Now we get

$$\begin{aligned}
i_F;\pi;e;\pi^{\smile};i_F &= i_F;i_F;\pi;e;\pi^{\smile};i_F;i_F \\
&= i_F;i_F^{\smile};\pi;e;\pi^{\smile};i_F;i_F \\
&\subseteq i_F;\rho;\Delta^{\smile};\pi^{\smile};\pi;e;\pi^{\smile};\pi;\Delta;\rho^{\smile};i_F \\
&\subseteq i_F;\rho;\Delta^{\smile};e;\Delta;\rho^{\smile};i_F \\
&\subseteq i_F;\rho;e^{\smile};\rho^{\smile};i_F
\end{aligned}$$

The converse inclusion follows analogously. □

$$\begin{tikzcd}
C \arrow[d, "c_F"] & \\
L^G \times L^M \arrow[loop right, "i_F"] & \\
L^G \arrow[loop left, "i_G"] \arrow[u, "\mathbb{I}\olessthan\Delta"'] \arrow[r, shift left, "\vartriangle"] & L^M \arrow[l, shift left, "\triangledown"] \arrow[ul, "\nabla\olessthan\mathbb{I}"'] \arrow[loop right, "i_M"] \\
D \arrow[u, "c_G"] & E \arrow[u, "c_M"']
\end{tikzcd}$$

Figure 4.1: A visual representation of the Lemma 3.3

The next lemma shows that set of concepts as pairs is isomorphic to the set of intends resp. extend. Not every bijective order-preserving map is an order-isomorphism.

In order to be isomorphic, the bijective order need to have an order-preserving inverse map [8].

Lemma 4.3. Let $c_F : C \to L^G \times L^M$ be the splitting of i_F, $c_G : D \to L^G$ be the splitting of i_G, $c_M : E \to L^M$ be the splitting of i_M. Then the relations

1. $c_M; (\nabla \circledless \mathbb{I}); c_F^{\smallsmile} : E \to C$,
2. $c_G; (\mathbb{I} \circledless \Delta); c_F^{\smallsmile} : D \to C$ are bijective maps.

Proof. 1) First we have

$$\begin{aligned}
&c_M; (\nabla \circledless \mathbb{I}); c_F^{\smallsmile}; c_F; (\nabla \circledless \mathbb{I})^{\smallsmile}; c_M^{\smallsmile} \\
&= c_M; (\nabla \circledless \mathbb{I}); i_F; (\nabla \circledless \mathbb{I})^{\smallsmile}; c_M^{\smallsmile} \\
&= c_M; (\nabla; \pi^{\smallsmile} \cap \rho^{\smallsmile}); (\pi; \Delta; \rho^{\smallsmile} \ \cap \ \rho; \nabla; \pi^{\smallsmile} \ \cap \ \mathbb{I}); (\pi; \nabla^{\smallsmile} \cap \rho); c_M^{\smallsmile} \\
&= c_M; ((\nabla; \pi^{\smallsmile} \cap \rho^{\smallsmile}); \pi; \Delta; \rho^{\smallsmile} \cap (\nabla; \pi^{\smallsmile} \cap \rho^{\smallsmile}); \rho; \nabla; \pi^{\smallsmile} \cap (\nabla; \pi^{\smallsmile} \cap \rho^{\smallsmile})); (\pi; \nabla^{\smallsmile} \cap \rho); c_M^{\smallsmile} \\
&= c_M; (\nabla; \Delta; \rho^{\smallsmile} \cap \nabla; \pi^{\smallsmile} \cap \nabla; \pi^{\smallsmile} \cap \rho^{\smallsmile}); (\pi; \nabla^{\smallsmile} \cap \rho); c_M^{\smallsmile} \\
&= c_M; (\nabla; \Delta; \rho^{\smallsmile} \cap \nabla; \pi^{\smallsmile} \cap \rho^{\smallsmile}); (\pi; \nabla^{\smallsmile} \cap \rho); c_M^{\smallsmile} \\
&= c_M; (\nabla; \Delta; \rho^{\smallsmile}; (\pi; \nabla^{\smallsmile} \cap \rho) \cap \nabla; \pi^{\smallsmile}; (\pi; \nabla^{\smallsmile} \cap \rho) \cap \rho^{\smallsmile}; (\pi; \nabla^{\smallsmile} \cap \rho)); c_M^{\smallsmile} \\
&= c_M; (\nabla; \Delta \cap \nabla; \Delta^{\smallsmile} \cap \mathbb{I}); c_M^{\smallsmile} \\
&= c_M; (\nabla; \Delta \cap \mathbb{I}); c_M^{\smallsmile} \\
&= c_M; c_M^{\smallsmile}; c_M; c_M^{\smallsmile} \\
&= \mathbb{I}.
\end{aligned}$$

For the opposite equation we first show

$$\begin{aligned}
&c_F; (\nabla \circledless \mathbb{I})^{\smallsmile}; c_M^{\smallsmile}; c_M; (\nabla \circledless \mathbb{I}); c_F^{\smallsmile} \\
&= c_F; (\pi; \nabla^{\smallsmile} \cap \rho); i_M; (\nabla; \pi^{\smallsmile} \cap \rho^{\smallsmile}); c_F^{\smallsmile} \\
&= c_F; (\pi; \nabla^{\smallsmile} \cap \rho); (\nabla; \Delta \cap \mathbb{I}); (\nabla; \pi^{\smallsmile} \cap \rho^{\smallsmile}); c_F^{\smallsmile} \\
&\subseteq c_F; (\pi; \nabla^{\smallsmile}; \nabla; \Delta \cap \pi; \nabla^{\smallsmile} \cap \rho); (\nabla; \pi^{\smallsmile} \cap \rho^{\smallsmile}); c_F^{\smallsmile} \\
&\subseteq c_F; (\pi; \nabla^{\smallsmile}; \nabla; \Delta; \nabla; \pi^{\smallsmile} \cap \pi; \nabla^{\smallsmile}; \nabla; \pi^{\smallsmile} \cap \rho; \nabla; \pi^{\smallsmile} \cap \pi; \nabla^{\smallsmile}; \nabla; \Delta; \rho^{\smallsmile} \cap \pi; \nabla^{\smallsmile}; \rho^{\smallsmile} \cap \rho; \rho^{\smallsmile}); c_F^{\smallsmile} \\
&\subseteq c_F; (\pi; \nabla^{\smallsmile}; \nabla; \pi^{\smallsmile} \cap \rho; \nabla; \pi^{\smallsmile} \cap \pi; \nabla^{\smallsmile}; \nabla; \Delta; \rho^{\smallsmile} \cap \rho; \rho^{\smallsmile}); c_F^{\smallsmile} \\
&\subseteq c_F; (\pi; \pi^{\smallsmile} \cap \rho; \nabla; \pi^{\smallsmile} \cap \pi; \nabla; \rho^{\smallsmile} \cap \rho; \rho^{\smallsmile}); c_F^{\smallsmile} \\
&= c_F; (\pi; \nabla; \rho^{\smallsmile} \cap \rho; \nabla; \pi^{\smallsmile} \cap \mathbb{I}); c_F^{\smallsmile} \\
&= c_F; I_F; c_F^{\smallsmile} \\
&= c_F; c_F^{\smallsmile}; c_F; c_F^{\smallsmile} \\
&= \mathbb{I},
\end{aligned}$$

and then

$$
\begin{aligned}
\mathbb{I} &= c_F; c_F^\smile; c_F; c_F^\smile \\
&= c_F; i_F; c_F^\smile \\
&= c_F; (\pi; \Delta; \rho^\smile \ \cap \ \rho; \nabla; \pi^\smile \ \cap \ \mathbb{I}); c_F^\smile \\
&= c_F; (\pi; \Delta; \rho^\smile \cap \rho; \nabla; \pi^\smile \cap \pi; \pi^\smile \cap \rho; \rho^\smile); c_F^\smile \\
&= c_F; (\pi; \Delta; \rho^\smile \cap \pi; \pi^\smile \cap \rho; (\nabla; \pi^\smile \cap \rho^\smile)); c_F^\smile \\
&\subseteq c_F; ((\pi; \Delta; \rho^\smile \cap \pi; \pi^\smile); (\pi; \nabla^\smile \cap \rho) \cap \rho); (\nabla; \pi^\smile \cap \rho^\smile); c_F^\smile \\
&= c_F; (\pi; \Delta \cap \pi; \nabla^\smile \cap \rho); (\nabla \olessthan \mathbb{I}); c_F^\smile \\
&\subseteq c_F; (\pi; \nabla^\smile \cap \rho); (\nabla; \pi^\smile \cap \rho^\smile); \pi; \Delta \cap \mathbb{I}); (\nabla \olessthan \mathbb{I}); c_F^\smile \\
&= c_F; (\nabla \olessthan \mathbb{I})^\smile; (\nabla; \Delta \cap \mathbb{I}); (\nabla \olessthan \mathbb{I}); c_F^\smile \\
&= c_F; (\nabla \olessthan \mathbb{I})^\smile; I_M(\nabla \olessthan \mathbb{I}); c_F^\smile \\
&= c_F; (\nabla \olessthan \mathbb{I})^\smile; c_M^\smile c_M; (\nabla \olessthan \mathbb{I}); c_F^\smile \\
&= c_F; (\nabla \olessthan \mathbb{I})^\smile; c_M^\smile; c_M; (\nabla \olessthan \mathbb{I}); c_F^\smile.
\end{aligned}
$$

2) is shown analogously.

□

All concepts generated from the context are ordered using a hierarchical relation which is called concept lattice. In the next definition we are defining the order on concepts based on pairs, on objects sets and on attributes sets.

Definition 4.3. With the notation introduced so far we define

1. $e_F = c_F; (\pi; e; \pi^\smile \cap \rho; e^\smile; \rho^\smile); c_F^\smile : C \to C$
2. $e_G = c_G; e; c_G^\smile : D \to D$
3. $e_M = c_M; e^\smile; c_M^\smile : E \to E$

The first relational formula in the above definition is the order on concepts, i.e., pairs of object and attribute sets. They are in relation iff the object sets are included $\pi; e; \pi^\smile$ and the attribute sets are in the opposite inclusion $\rho; e; \rho^\smile$. In the second formula e_G is true for sets of objects that are part of a concept iff the first is included in the second. The last formula is basically the same as second but for attribute sets and the inclusion order is reversed here.

In the above definition, we used both object and attribute set from the concept in order to define the lectic order. In the next lemma, we are going to show that the order on pairs is determined by the order on one of its components only.

Lemma 4.4. $e_F = c_F;\pi;e;\pi^{\smile};c_F^{\smile} = c_F;\rho;e^{\smile};\rho^{\smile};c_F^{\smile}$

Proof. We only show $e_F = c_F;\rho;e^{\smile};\rho^{\smile};c_F^{\smile}$. The other equation follows analogously.

$$\begin{aligned}
e_F &= c_F;(\pi;e;\pi^{\smile} \cap \rho;e^{\smile};\rho^{\smile});c_F^{\smile} \\
&= c_F;\pi;e;\pi^{\smile};c_F^{\smile} \cap c_F;\rho;e^{\smile};\rho^{\smile};c_F^{\smile} \\
&= c_F;c_F^{\smile};c_F;\pi;e;\pi^{\smile};c_F^{\smile};c_F;c_F^{\smile} \cap c_F;\rho;e^{\smile};\rho^{\smile};c_F^{\smile} \\
&= c_F;i_F;\pi;e;\pi^{\smile};i_F;c_F^{\smile} \cap c_F;\rho;e^{\smile};\rho^{\smile};c_F^{\smile} \\
&= c_F;i_F;\rho;e^{\smile};\rho^{\smile};i_F;c_F^{\smile} \cap c_F;\rho;e^{\smile};\rho^{\smile};c_F^{\smile} && \text{Lemma 3.2} \\
&= c_F;c_F^{\smile};c_F;\rho;e^{\smile};\rho^{\smile};c_F^{\smile};c_F;c_F^{\smile} \\
&= c_F;\rho;e^{\smile};\rho^{\smile};c_F^{\smile}
\end{aligned}$$

□

Last but not least, we show that the bijections from Lemma 3.3 are monotonic. We only state this result for the first bijection. The second monotonicity property is again shown analogously.

Lemma 4.5. Let $e_F : C \to C$ and $e_M : E \to E$ are two order relation. The bijection relation $c_M;(\nabla \otimes \mathbb{I});c_F^{\smile} : E \to C$ is called monotone iff $c_M;(\nabla \otimes \mathbb{I});c_F^{\smile};e_F \subseteq e_M;c_M;(\nabla \otimes \mathbb{I});c_F^{\smile}$ or equivalently $e_M \subseteq c_M;(\nabla \otimes \mathbb{I});c_F^{\smile};e_F^{\smile};(c_M;(\nabla \otimes \mathbb{I});c_F^{\smile})^{\smile}$.

Proof. First, we have

$$\begin{aligned}
&c_M;(\nabla \otimes \mathbb{I});c_F^{\smile};e_F^{\smile};(c_M;(\nabla \otimes \mathbb{I});c_F^{\smile})^{\smile} \\
&= c_M;(\nabla \otimes \mathbb{I});c_F^{\smile};e_F^{\smile};c_F;(\nabla \otimes \mathbb{I})^{\smile};c_M^{\smile} \\
&= c_M;(\nabla \otimes \mathbb{I});c_F^{\smile};c_F;\rho;e^{\smile};\rho^{\smile};c_F^{\smile};c_F;(\nabla \otimes \mathbb{I})^{\smile};c_M^{\smile} && \text{Lemma 3.2} \\
&\subseteq c_M;(\nabla \otimes \mathbb{I});\rho;e^{\smile};\rho^{\smile};(\nabla \otimes \mathbb{I})^{\smile};c_M^{\smile} \\
&= c_M;(\nabla \otimes \mathbb{I});\rho;e^{\smile};((\nabla \otimes \mathbb{I});\rho)^{\smile};c_M^{\smile} \\
&= c_M;e^{\smile};c_M^{\smile} \\
&= e_M.
\end{aligned}$$

This implies the assertion.

□

4.2 Attribute Implication

In this section, we want to focus on attribute implications. As mentioned before, attribute implications summarize properties of the given data in a way that can easily be understood by a human. One of the main purposes of this section is to construct and prove the algebraic formulas for generating a set of all attribute implications from a Boolean context as well as a multi-valued context. Another goal is to define a recursive function of implication basis from the context since sometimes we are only interested in a small set of implications which followed by all other valid implications in the context. Then we proposed a relational algebraic formula using the implication basis to generate a valid implication set satisfying certain rules of all implications.

Definition 4.4. An attribute implication is an expression $P \to Q$, where $P, Q \subseteq M$, is true in $\mathbb{K}$ if each object which has all attributes from P has also all attributes from Q.

We now want to express attribute implications algebraically. In order to do so we define a relation M_I so that in the case of classical relations, $M_I(A, B)$ is true iff $A \to B$ is an implication. This means that if $A \subseteq T$ then $B \subseteq T$ for all sets of intents of formal concepts.

Definition 4.5. A subset $T \subseteq M$ respects an implication $A \to B$ if $A \subseteq T$ or $B \nsubseteq T$ i.e. $A \subseteq T \to B \subseteq T$. T respects a set M_I of implications if T respects every single implication in M_I. We can define the set of implication M_I algebraically as follow

$$M_I = c_M; e^{\smile} \backslash c_M; e^{\smile}$$

Here, the relation M_I is an embedding between all attribute subsets from the context. The first part of the formula i.e. $c_M; e^{\smile}$ generates a relation from attribute sets which are concept to all attribute sets. This relation is true iff an attribute set is included in the attribute concept. The final relation M_I is true for those attribute sets which maintain a valid implication in respect to the formal context which supports all attribute concepts in c_M. We can easily construct all valid implications by traversing the generated relation from the above formula .

The relation M_I can alternatively computed by the following lemma. The purpose of this lemma is to construct and prove an alternative form of relational algebra for producing all valid implications from the context. The composition operator $\nabla; \Delta$

has been used here for generating the attribute concepts as well as all attribute implications.

Lemma 4.6. $M_I = ((\nabla;\Delta\cap\mathbb{I});e^{\smallsmile})\backslash e^{\smallsmile}$

Proof.

$$\begin{aligned} X \subseteq M_I = c_M;e^{\smallsmile}\backslash c_M;e^{\smallsmile} \\ &\Leftrightarrow c_M;e^{\smallsmile};X \subseteq c_M;e^{\smallsmile} \\ &\Leftrightarrow c_M^{\smallsmile};c_M;e^{\smallsmile};X \subseteq e^{\smallsmile} \\ &\Leftrightarrow (\nabla;\Delta\cap\mathbb{I});e^{\smallsmile};X \subseteq e^{\smallsmile} \\ &\Leftrightarrow X \subseteq ((\nabla;\Delta\cap\mathbb{I});e^{\smallsmile})\backslash e^{\smallsmile} \end{aligned}$$

□

Another way of obtaining M_I is given by the next lemma.

Lemma 4.7. An implication $P \to Q$ holds in (G,M,I) if and only if $Q \subseteq P^{\nabla\Delta}$. Relation algebraically we have.

$$M_I = \nabla;\Delta;e^{\smallsmile}$$

Proof. First, we show $(\nabla;\Delta\cap\mathbb{I});e^{\smallsmile};\nabla;\Delta;e^{\smallsmile};\varepsilon^{\smallsmile} \subseteq \varepsilon^{\smallsmile}$

$$\begin{aligned} &(\nabla;\Delta\cap\mathbb{I});e^{\smallsmile};\nabla;\Delta;e^{\smallsmile};\varepsilon^{\smallsmile} \\ &\subseteq (\nabla;\Delta\cap\mathbb{I});e^{\smallsmile};\nabla;\Delta;\varepsilon^{\smallsmile} \\ &\subseteq (\nabla;\Delta\cap\mathbb{I});\nabla;\Delta;e^{\smallsmile};\varepsilon^{\smallsmile} && \text{Theorem 4.1 of [3]} \\ &\subseteq (\nabla;\Delta\cap\mathbb{I});\nabla;\Delta;\varepsilon^{\smallsmile} \\ &= (\nabla;\Delta)^{\smallsmile};\nabla;\Delta;\nabla;\Delta;\varepsilon^{\smallsmile} && \text{Lemma 3.5(a) of [3]} \\ &= (\nabla;\Delta)^{\smallsmile};\nabla;\Delta;\varepsilon^{\smallsmile} && \text{Theorem 4.1 of [3]} \\ &= (\nabla;\Delta\cap\mathbb{I});\varepsilon^{\smallsmile} && \text{Lemma 3.5 of [3]} \\ &\subseteq \varepsilon^{\smallsmile} \end{aligned}$$

□

This implies $(\nabla;\Delta\cap\mathbb{I});e^{\smallsmile};\Delta;\nabla;e^{\smallsmile} \subseteq \varepsilon^{\smallsmile}/\varepsilon^{\smallsmile} = e^{\smallsmile}$, and, hence $\Delta;\nabla;e^{\smallsmile} \subseteq (\nabla;\Delta\cap\mathbb{I});e^{\smallsmile}\backslash e^{\smallsmile}$. From Lemma 3.6 we conclude $\Delta;\nabla;e^{\smallsmile} \subseteq M_I$. For the opposite inclusion we show $(\nabla;\Delta)^{\smallsmile};((\nabla;\Delta\cap\mathbb{I});e^{\smallsmile}\backslash e^{\smallsmile}) \subseteq e^{\smallsmile}$. Consider the following computation

$$(\nabla;\Delta)^{\smallsmile};((\nabla;\Delta\cap\mathbb{I});e^{\smallsmile}\backslash e^{\smallsmile})$$

$$
\begin{aligned}
&= (\nabla;\Delta)^{\smile};\nabla;\Delta;(\nabla;\Delta)^{\smile};((\nabla;\Delta \cap \mathbb{I});e^{\smile}\backslash e^{\smile}) && \\
&= (\nabla;\Delta \cap \mathbb{I});(\nabla;\Delta)^{\smile};((\nabla;\Delta \cap \mathbb{I});e^{\smile}\backslash e^{\smile}) && \text{Lemma 3.5(a) of [3]} \\
&\subseteq (\nabla;\Delta \cap \mathbb{I});e^{\smile};((\nabla;\Delta \cap \mathbb{I});e^{\smile}\backslash e^{\smile}) && \text{Theorem 4.1 of [3]} \\
&\subseteq e^{\smile}
\end{aligned}
$$

From this we get, $M_I = (\nabla;\Delta \cap \mathbb{I});e^{\smile}\backslash e^{\smile} \subseteq \nabla;\Delta;e^{\smile}$. The ultimate relation generated from this lemma is true for an attribute set iff it is included in the closure of an attribute set from the context.

Normally, the set of implications of a given context is huge. Furthermore, some implications seem to follow easily from others. For those reasons, we are usually only interested in the basis of implications, i.e., a set of implications that will imply using certain simple rules all implications.

In this thesis, we are interested in the so-called Duquenne-Guigues or canonical basis [8]. To define it, we need to denote some basic definition as follow.

Definition 4.6. A set $A \subseteq M$ is called pseudo-closed iff it satisfies the following two conditions.

1. $A \neq A^{\nabla\Delta}$
2. For every pseudo-closed $B \subseteq A$ we have $B^{\nabla\Delta} \subseteq A$

The definition above is actually recursive due to the reference to pseudo-closed sets in Property 2. The base case of the recursion is given by sets that satisfy 1. and do not contain any set of the form $B^{\nabla\Delta}$. The Duquenne-Guigues basis is now given as $\mathcal{I} := \{P \rightarrow P^{\nabla\Delta} \mid P \text{ is pseudo-closed}\}$. Finally we present formulas for computing the pseudo-closed sets relation algebraically.

We start with sets that do not contain and intend of a concept. Lets define the above propositions algebraically as below

$$
\begin{aligned}
a &= [(e \cap \overline{\mathbb{I}})\backslash c_M^{\smile};\top\!\!\top] \cap \overline{c_M^{\smile};\top\!\!\top} \\
next(a) &= a \cup [((e \cap \overline{\mathbb{I}} \cap a;\top\!\!\top)\backslash \nabla;\Delta;e) \cap \mathbb{I});\top\!\!\top \cap \overline{c_M^{\smile};\top\!\!\top}]
\end{aligned}
$$

In the definition of a, the part before intersection is true for a set of attributes if every strictly smaller set is not a concept. The second part says it is not a concept itself i.e. not closed set. Then we input this initial result to the function *next* which generates all pseudo-closed sets satisfying the same conditions by the second part of the union and merges the new result with the previous one. We need to iterate the above function until nothing new and then finally we can determine our canonical base B as follow.

First, let $\mathcal{I}$ be the fixed point of next starting from a and applying next. Then

$$B = \mathcal{I}; \pi \cap \nabla; \Delta$$

The first part of the intersection in the above formula generates a relation that contains all 1 in the row of all pseudo attribute sets from $\mathcal{I}$. The next part of the intersection produces closure of all attribute sets and finally, the union operation results in true for the pseudo-closed attribute set and their corresponding closure set. The relation B eventually holds all implication of the form $P \rightarrow P^{\nabla\Delta}$ where P is pseudo-closed attribute set.

Now we can use the implication basis B to generate all valid implications from the formal context. In the next proposition, we stated all the required conditions need to generate all implication sets using the implication basis.

Proposition 4.1. For all $A, B, C \subseteq M$, we can produce all the implications recursively satisfying following three conditions

1. If $B \subseteq A$ then $A \rightarrow B$
2. If $A \rightarrow B$ then $A \cup C \rightarrow B \cup C$
3. If $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$

We do not do a proof of the above proposition here

We can define a function which takes implication basis as input and recursively generates implications until nothing new as follow

$$\text{Comp(B)} = e^{\smile} \cup J^{\smile}; (\pi; B; \pi^{\smile} \cap \rho; \rho^{\smile}); J \cup \overline{B; B} \cup B$$
$$J = syq(\varepsilon; \pi^{\smile} \cup \varepsilon; \rho^{\smile}, \varepsilon)^{\downarrow}$$

The first part in the above formula i.e. $e^{\smile}$ satisfies the first condition in the proposition 4.1. The second part and third part of the union operation corresponds to the next two conditions respectively in the proposition. Ultimately, we can generate all implications from the implication basis B using the function defined above.

Chapter 5

Implementation

5.1 General Information

5.1.1 System Overview

The FCA suite is an application that utilizes an intuitive user interface which makes generating concepts and attributes implication easier. This program provides a graphical user interface that allows a user to input formal context and store results electronically in excel format for further analysis.

This program takes objects, attributes and lattice information in the form of XML. Then it dynamically generates a table with rows and columns equal to the number of objects and attributes respectively. This table hold the formal context which is later used for generating concepts and attribute implications.

5.1.2 System Description

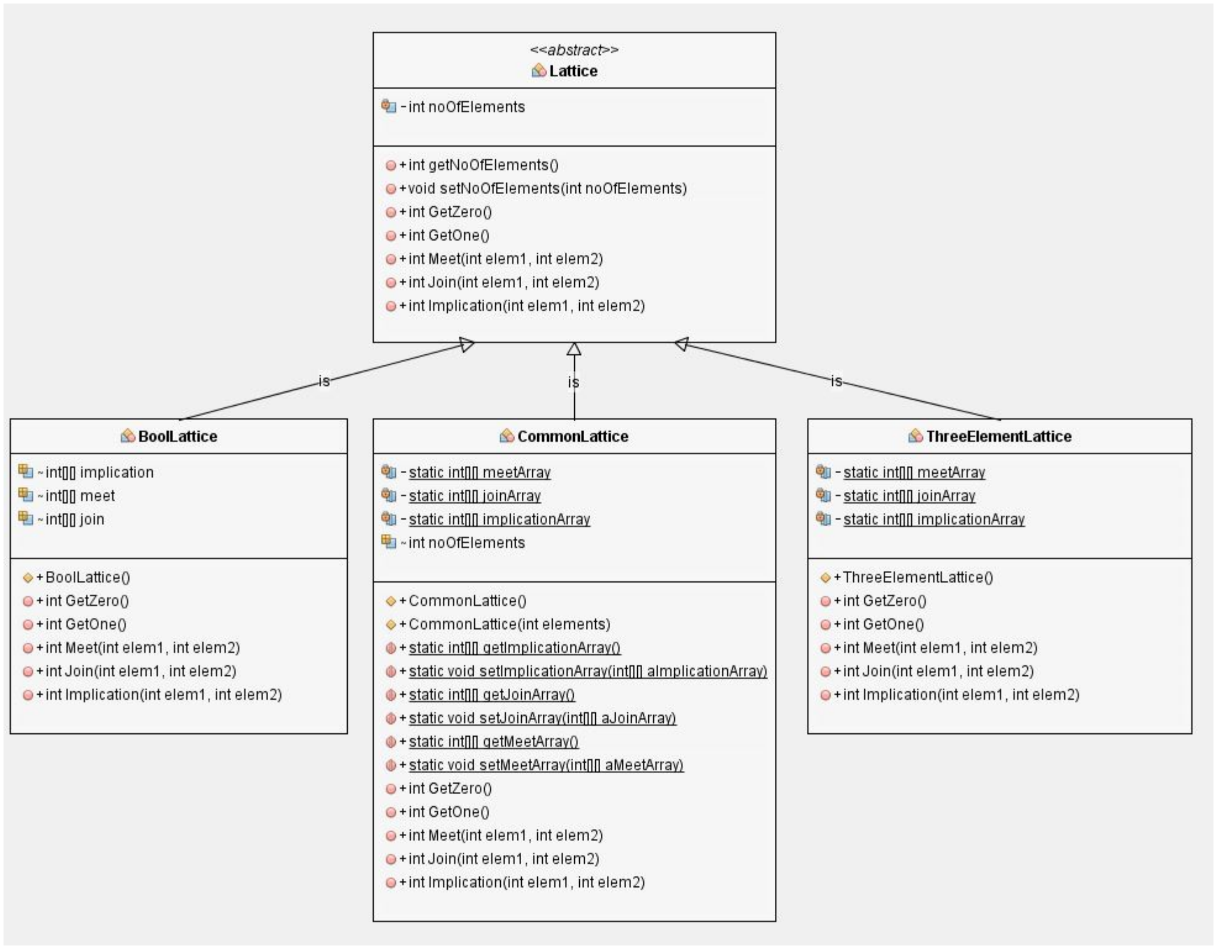
We implemented a base project for testing and proofing algebraic formulas related to the L-fuzzy formal concept analysis stated in the previous chapter. The project is divided into multiple packages with java classes. There are a total of five packages in our project where two packages related to lattice and their operations, another two related to GUI and the other one contains the utility class. Here we have summarized only the basic methods related to lattice and their operations.

com.fca.lattice

This package holds the classes for representing lattice element with their associated operations. Their is an abstract super-class named Lattice inherited by several class shown in the class diagram (Figure 5.1). The list of basic methods and description has been summarized in Table 5.1 below.

Modifier and Type	**Method and Description**
int	**getNoOfElements()** This method helps to access the private field noOfElements.
void	**setNoOfElements(int noOfElements)** This method sets the value to the private field noOfElements.
abstract int	**GetOne()** This method returns the top element of lattice.
abstract int	**GetZero()** This method returns the bottom element of the lattice.By Default, 0 is the value element of any lattice.
abstract int	**Implication(int elem1, int elem2)** This method determines the greatest element (x) such that the meet of x and elem1 is less or equal to elem2.
abstract int	**Join(int elem1, int elem2)** This method determines the join or supremum of two elements.In other words, it returns the least element from the set of elements where each element of that set is greater than both input variables.
abstract int	**Meet(int elem1, int elem2)** This method determines the meet or infimum of two elements.

Table 5.1: Brief overview of lattice package



<<abstract>>
Lattice
- int noOfElements
+ int getNoOfElements()
+ void setNoOfElements(int noOfElements)
+ int GetZero()
+ int GetOne()
+ int Meet(int elem1, int elem2)
+ int Join(int elem1, int elem2)
+ int Implication(int elem1, int elem2)
is
is
is
BoolLattice
~ int[][] implication
~ int[][] meet
~ int[][] join
+ BoolLattice()
+ int GetZero()
+ int GetOne()
+ int Meet(int elem1, int elem2)
+ int Join(int elem1, int elem2)
+ int Implication(int elem1, int elem2)
CommonLattice
- static int[][] meetArray
- static int[][] joinArray
- static int[][] implicationArray
~ int noOfElements
+ CommonLattice()
+ CommonLattice(int elements)
+ static int[][] getImplicationArray()
+ static void setImplicationArray(int[][] aImplicationArray)
+ static int[][] getJoinArray()
+ static void setJoinArray(int[][] aJoinArray)
+ static int[][] getMeetArray()
+ static void setMeetArray(int[][] aMeetArray)
+ int GetZero()
+ int GetOne()
+ int Meet(int elem1, int elem2)
+ int Join(int elem1, int elem2)
+ int Implication(int elem1, int elem2)
ThreeElementLattice
- static int[][] meetArray
- static int[][] joinArray
- static int[][] implicationArray
+ ThreeElementLattice()
+ int GetZero()
+ int GetOne()
+ int Meet(int elem1, int elem2)
+ int Join(int elem1, int elem2)
+ int Implication(int elem1, int elem2)

Figure 5.1: Class Diagram of lattice package

com.fca.operations

This package holds the classes for representing operations for generating formal concepts and attribute implications. Here we summarized the main methods with their description in Table 5.2.

Modifier and Type	**Method and Description**
Relation	**union(Relation a)** This method determines the union of two relations with same dimension. the output relation.
Relation	**intersection(Relation a)** This method determines the union of two relations with same dimension.
Relation	**implication(Relation a)** This method determines the implication of two relation with same dimension.
Relation	**composition(Relation a)** This method calculates the relative multiplication of two relation.
Relation	**symmetricQuotient(Relation a)** This method calculates the intersection of left residue and right residue to two relation R and S.
Relation	**transpose(Relation a)** This method flips the relation, that is it switches the row and column by producing another relation.
Relation	**leftResidue(Relation a)** This method calculates the right residual of two relation R and S which is the greatest of all relation X with $(X.R)$ is the subset of S.
Relation	**rightResidue(Relation a)** This method calculates the left residual of two relation R and S $(R\backslash S)$ which is the greatest of all relation X with $(R.X)$ is the subset of S.
Relation	**up()** This function takes relation as a input. If the relation entry getter than 0 then this makes the entry 1 otherwise keep the same.
Relation	**down()** This function takes relation as a input. If the relation entry equal to 1 then this makes the entry 1 otherwise keep the same.
Relation	**equivalence()** This method returns the top element of lattice.

Table 5.2: Brief overview of operation package

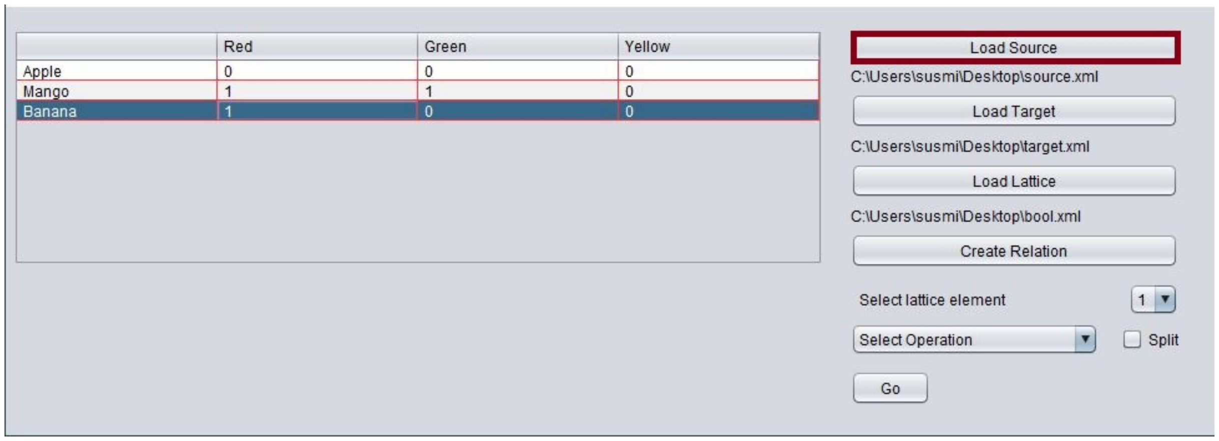
Modifier and Type	Method and Description
Relation	**generateFullConcept()** This method generates concepts satisfying the condition $A^{\Delta} = B$ and $B^{\nabla} = A$. The generated concepts in the form of relation where column and row represent object and attribute subset respectively.
Relation	**generateObjectConcept()** This method generates concepts$(A^{\Delta\nabla}, A^{\Delta})$ satisfying the condition $A^{\Delta\nabla} = A$. The generated concepts in the form of relation where column and row represent object and attribute subset respectively.
Relation	**generateAttributeConcept()** This method generates concepts$(B^{\nabla\Delta}, B^{\nabla})$ satisfying the condition $B^{\nabla\Delta} = B$. The generated concepts in the form of relation where column and row represent object and attribute subset respectively.
Relation	**generateCanonicalBase()** This method produces the implication basis of the context.
Relation	**generateImplicationUsingLemma()** This function generates all the implications using lemma.
Relation	**generateImplicationsUsingBasis()** This methods generates all the implication using implication basis.

Table 5.3: Brief overview of operation package

5.2 User Manual

5.2.1 Load source and target

Firstly, the user needs to input the source and target information in the XML format shown in Listing 5.1 and 5.2 respectively. The title tag represents the object/target name which will show as row and column name of the table in the GUI. There are two clickable buttons marked on the GUI shown in Figure 5.2 which will redirect the user to browse the source and target XML file respectively. The program can not handle any other file format except XML.



	Red	Green	Yellow
Apple	0	0	0
Mango	1	1	0
Banana	1	0	0

Load Source
C:\Users\susmi\Desktop\source.xml
Load Target
C:\Users\susmi\Desktop\target.xml
Load Lattice
C:\Users\susmi\Desktop\bool.xml
Create Relation
Select lattice element
1
Select Operation
Split
Go

Figure 5.2: GUI for loading source and target

```
<?xml version="1.0" encoding="UTF-8"?>
<source>
  <title>Apple</title>
  <title>Mango</title>
  <title>Banana</title>
</source>
```

Listing 5.1: XML format for source

```
<?xml version="1.0" encoding="UTF-8"?>
<target>
  <title>Red</title>
  <title>Green</title>
  <title>Yellow</title>
</target>
```

Listing 5.2: XML format for target

5.2.2 Load lattice

The lattice information must also be submitted in the form of XML shown in listing 5.3. The meet, join and implication tag represent the element-wise meet, join and implication value which will be used for generating concepts and attribute implications. The value should be placed row-wise separated by a comma in the corresponding tag.

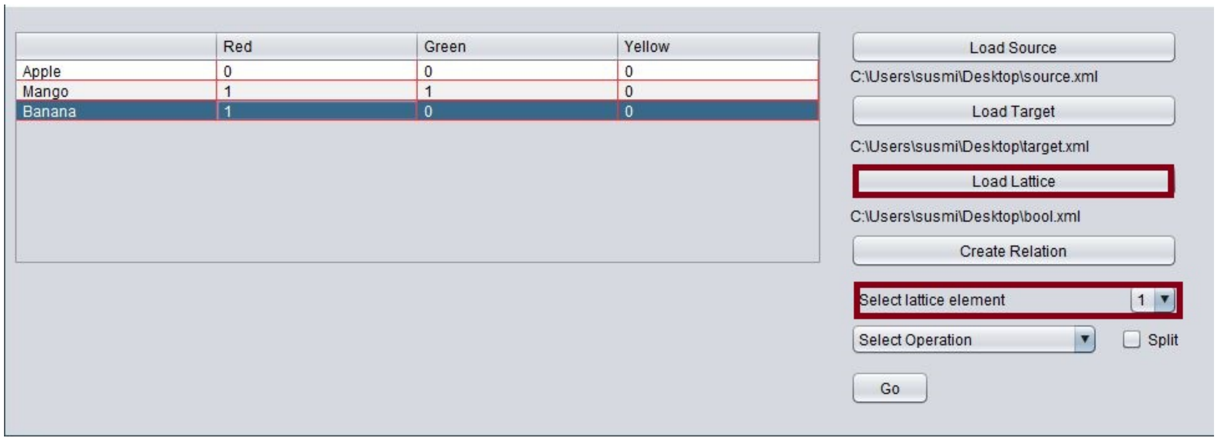
Lets explain the process of making XML to represent the lattice. The meet, join and implication of Boolean lattice can be shown by a two-dimensional array as follow.

$$Meet = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array} \qquad Join = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 1 \end{array} \qquad Implication = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1 & 1 \\ 1 & 0 & 1 \end{array}$$

Now we need to take the values from the matrix to the corresponding tag in XML and should be placed row-wise separated by a comma. The Listing 5.3 represents the data from the above matrices. For example, the meet tag holds the value from the Meet matrix and the values are placed row-wise. The process is similar to join and implication.

```
<?xml version="1.0" encoding="UTF-8"?>
<lattice>
<elements>2</elements>
<meet>0,0,0,1</meet>
<join>0,1,1,1</join>
<implication>1,1,0,1</implication>
</lattice>
```

Listing 5.3: XML format for Boolean lattice



	Red	Green	Yellow
Apple	0	0	0
Mango	1	1	0
Banana	1	0	0

C:\Users\susmi\Desktop\source.xml
C:\Users\susmi\Desktop\target.xml
C:\Users\susmi\Desktop\bool.xml
Select lattice element: 1

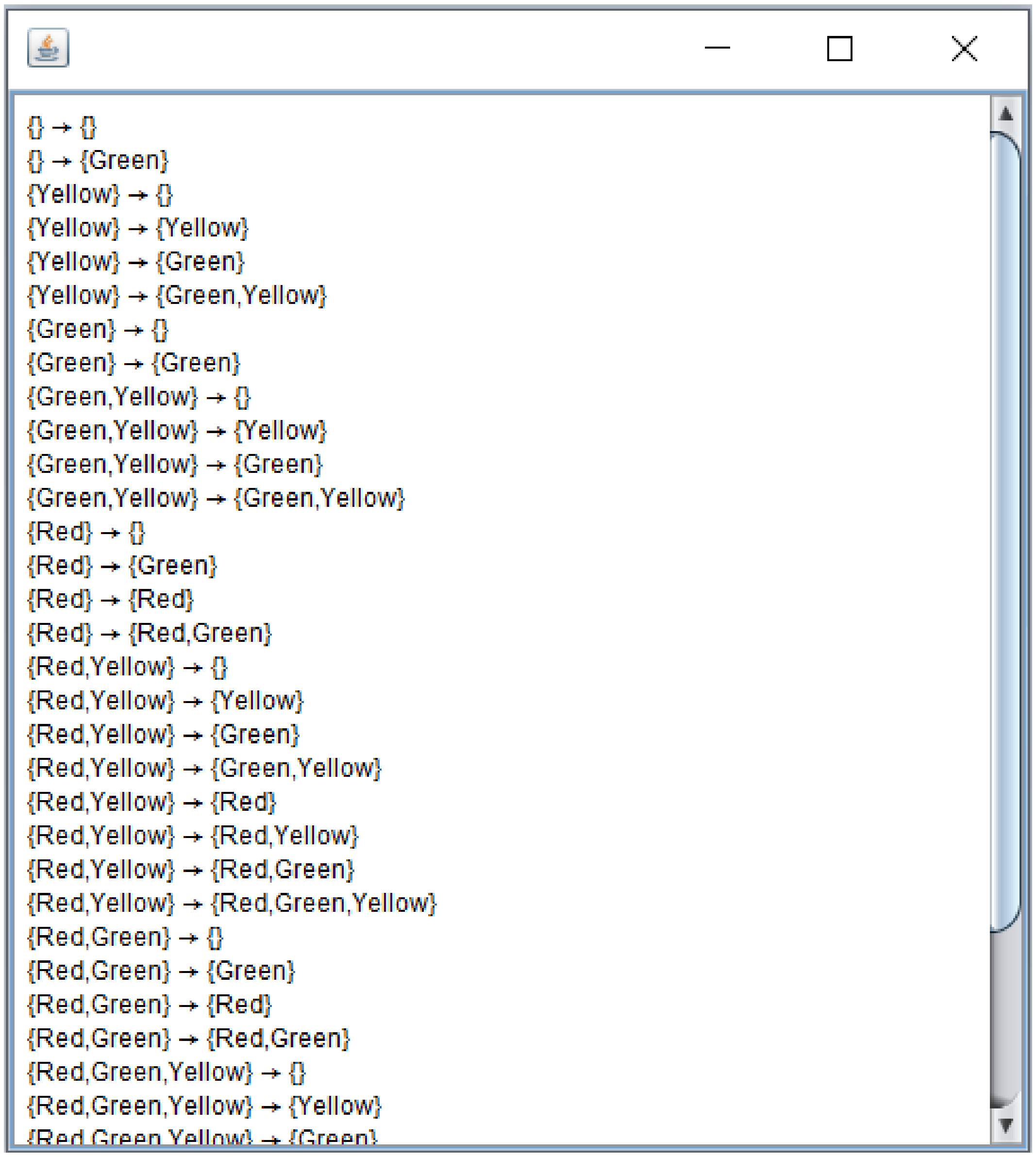
Figure 5.3: GUI for creating relation

5.2.3 Create relation

After giving the source, target and lattice information, we are ready to generate the context in the form of a table. We can fill-up the table by selecting the element from the combo box at the lower right corner which represents the lattice elements.

Select operation

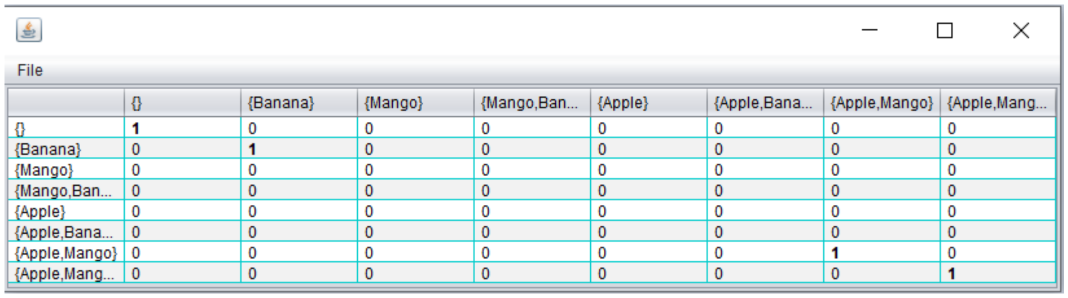
There is a total of five operations related to concept analysis. We can generate intent concepts, extent concepts, full concepts, implications base and all implications using implication basis. The split operation removes those rows from the result which has no entry greater than zero. Generated concepts will be displayed in a different window shown in Figure 5.5. They are presented in the form of a table where rows and columns represent the object and attribute subset respectively. The nonzero entry in the diagonal represents the degree of concept while row and column are the concept extent and intent respectively.



```
{} → {}
{} → {Green}
{Yellow} → {}
{Yellow} → {Yellow}
{Yellow} → {Green}
{Yellow} → {Green,Yellow}
{Green} → {}
{Green} → {Green}
{Green,Yellow} → {}
{Green,Yellow} → {Yellow}
{Green,Yellow} → {Green}
{Green,Yellow} → {Green,Yellow}
{Red} → {}
{Red} → {Green}
{Red} → {Red}
{Red} → {Red,Green}
{Red,Yellow} → {}
{Red,Yellow} → {Yellow}
{Red,Yellow} → {Green}
{Red,Yellow} → {Green,Yellow}
{Red,Yellow} → {Red}
{Red,Yellow} → {Red,Yellow}
{Red,Yellow} → {Red,Green}
{Red,Yellow} → {Red,Green,Yellow}
{Red,Green} → {}
{Red,Green} → {Green}
{Red,Green} → {Red}
{Red,Green} → {Red,Green}
{Red,Green,Yellow} → {}
{Red,Green,Yellow} → {Yellow}
{Red,Green,Yellow} → {Green}
```

Figure 5.4: GUI for representing attribute implications

There is an option in the menu bar for saving the result in the excel format. Implications are shown on two separate windows, one represents implications in tabular form and another summarizes the implications in the form of $P \rightarrow Q$ as shown in Figure 5.4. We can also save the implications in excel format.



File

	{}	{Banana}	{Mango}	{Mango,Ban...	{Apple}	{Apple,Bana...	{Apple,Mango}	{Apple,Mang...
{}	**1**	0	0	0	0	0	0	0
{Banana}	0	**1**	0	0	0	0	0	0
{Mango}	0	0	0	0	0	0	0	0
{Mango,Ban...	0	0	0	0	0	0	0	0
{Apple}	0	0	0	0	0	0	0	0
{Apple,Bana...	0	0	0	0	0	0	0	0
{Apple,Mango}	0	0	0	0	0	0	**1**	0
{Apple,Mang...	0	0	0	0	0	0	0	**1**

Figure 5.5: GUI for representing concepts

Chapter 6

Conclusion

In this thesis, we have proposed a general framework for generating formal concepts and attribute implications from an L-fuzzy formal context. Normally Formal Concept Analysis (FCA) handles multi-valued context by applying conceptual scaling. Our proposed algebraic approach can process a multi-valued context without any scaling i.e. keeping the data as-is. One of the advantages of our approach is that the generic relation formulas in our framework can be used to process Boolean as well as Fuzzy context. Another benefit of our model is that there is no chance of biasness in the experimental result. We have seen FCA deals with fuzzy context by applying conceptual scaling but there is logical explanation behind the selection of scale attributes to pre-process the data. In the example of conceptual scaling, we considered the age range 15-24 years as young but there is no logical reason supporting this selection. The experimental results will change according with the variation in the user selection of scale attributes. In contrast, our proposed framework deals with fuzzy context with any data pre-processing which results in unbiased experimental outcome.

We can generate interesting patterns from the data source in the form of association rules or attribute implication using our proposed framework. The advantage of our solution is that it can generate a non-redundant implication basis as well as all valid implications in the context. In the case of a fuzzy formal context, the total number of implications would be enormous and difficult to read all implications. That's why we are sometime interested in implication basis which we can generate all implications.

Currently, our implementation is unable to handle a large context. For a set S, a context of the form (S, S) will be called a contranominal scale. In case of contranominal scale with m objects (and m attributes), the maximum number of concept extent and concept intent would be 2^m. Our proposed solution takes exponential time in the worst case to generate all concepts from a formal context which is a fact for a large formal context. The time complexity for generating all concept extents and intents is $O(|G|^2|M|)$ and $O(|G||M|^2)$ respectively where G and M are object and attribute set respectively. We are using our implementation to prove the concepts of formal concept analysis. Future work will focus on improving the computational complexity of our proposed formulas. After improving the computational complexity, we will able to process large context as well. We hope that our work will contribute to expanding the data mining research using L-fuzzy formal concept analysis.

Bibliography

[1] Radim Belohlavek. What is a fuzzy concept lattice? ii. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, pages 19–26. Springer, 2011.

[2] Radim Belohlavek and Jan Konecny. Scaling, granulation, and fuzzy attributes in formal concept analysis. In *2007 IEEE International Fuzzy Systems Conference*, pages 1–6. IEEE, 2007.

[3] Rudolf Berghammer and Michael Winter. Decomposition of relations and concept lattices. *Fundamenta Informaticae*, 126(1):37–82, 2013.

[4] Peter Burmeister and Richard Holzer. Treating incomplete knowledge in formal concept analysis. In *Formal Concept Analysis*, pages 114–126. Springer, 2005.

[5] Paolo Ceravolo, E Damiani, and M Viviani. Extending formal concept analysis by fuzzy bags. *Proceedings of IPMU, Paris*, 2006.

[6] Yassine Djouadi and Henri Prade. Interval-valued fuzzy formal concept analysis. In *International Symposium on Methodologies for Intelligent Systems*, pages 592–601. Springer, 2009.

[7] Hitoshi Furusawa and Wolfram Kahl. *A study on symmetric quotients*. Univ. der Bundeswehr, Fak. für Informatik, 1998.

[8] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.

[9] Tzung-Pei Hong and Yeong-Chyi Lee. An overview of mining fuzzy association rules. In *Fuzzy Sets and Their Extensions: Representation, Aggregation and Models*, pages 397–410. Springer, 2008.

[10] Nurbek Imangazin. A relation-algebraic approach to l-fuzzy topology. 2019.

[11] Ch Aswani Kumar. Fuzzy clustering-based formal concept analysis for association rules mining. *Applied artificial intelligence*, 26(3):274–301, 2012.

[12] Gunther Schmidt. *Relational mathematics*, volume 132. Cambridge University Press, 2011.

[13] Michael Winter. Relational constructions in goguen categories. In *International Conference on Relational Methods in Computer Science*, pages 212–227. Springer, 2001.

[14] Michael Winter. *Goguen Categories. A Categorical Approach to L-fuzzy Relations.* Trends in Logic 25. Springer, 2007.

[15] Michael Winter. Arrow categories. 2009.

[16] Michael Winter. *T-Norm based Operations in Arrow Categories.* Relational and Algebraic Methods in Computer Science,LNCS 11194:70-86, 2017.

[17] Michael Winter. Type-n arrow categories. In *International Conference on Relational and Algebraic Methods in Computer Science*, pages 307–322. Springer, 2017.