Relational Approach to the $\mathcal{L}$-Fuzzy Concept Analysis

Anahita Izadpanahi

Submitted in partial fulfilment
of the requirements for the degree of

Master of Science

Department of Computer Science
Brock University
St. Catharines, Ontario

Abstract

Modern industrial production systems benefit from the classification and processing of objects and their attributes. In general, the object classification procedure can coincide with vagueness. Vagueness is a common problem in object analysis that exists at various stages of classification, including ambiguity in input data, overlapping boundaries between classes or regions, and uncertainty in defining or extracting the properties and relationships of objects. To manage the ambiguity mentioned in the classification of objects, using a framework for $\mathcal{L}$-fuzzy relations, and displaying such uncertainties by it can be a solution. Obtaining the least unreliable and uncertain output associated with the original data is the main concern of this thesis.

Therefore, my general approach to this research can be categorized as follows:

We developed a $\mathcal{L}$-*Fuzzy Concept Analysis* as a *generalization* of a *regular Concept Analysis*. We start our work by providing the input data. Data is stored in a table (database). The next step is the creation of the contexts and concepts from the given original data using some structures. In the next stage, rules or patterns (Attribute Implications) from the data will be generated. This includes all rules and a minimal base of rules. All of them are using $\mathcal{L}$-fuzziness due to uncertainty. This requires $\mathcal{L}$-fuzzy relations that will be implemented as $\mathcal{L}$-valued matrices. In the end, everything is nicely packed in a convenient application and implemented by *Java* programming language. Generally, our approach is done in an *algebraic framework* that covers both *regular* and $\mathcal{L}$-*Fuzzy FCA*, simultaneously.

The tables we started with are already $\mathcal{L}$-valued (not crisp) in our implementation. In other words, we work with the $\mathcal{L}$-Fuzzy data directly. This is the idea here. We start with vague data.

In simple terms, the data is shown using $\mathcal{L}$-valued tables (vague data) trying to relate objects with their attributes at the start of the implementation. Generating attribute implications from many-valued contexts by a relational theory is the purpose of this thesis, i.e, a range of degrees is used to indicate the relationship between objects and their properties. The

smallest degree corresponds to the classical no and the greatest degree corresponds to the classical yes in the table.

Acknowledgements

First of all, I want to appreciate Dr. Michael Winter for his key role in achieving my life aspirations. I am very proud to learn more and more from him. His knowledge, constant support, illuminating Comments, advice, and more important his understanding of the lonely life of an international student and kindness motivated me to pass this critical stage of my educational life. His patience will be the beacon of my life. He has changed my life.
My husband or in a better word my life, who has never and under no circumstances withheld his support from me.
My parents who have always paved the way for my dreams in life; from childhood until now.
My sisters who have always tried to support me at every stage of my life, just like someone who gives you his experiences in life without any expectations.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays data-sets are available in very complex and heterogeneous ways. Mining of such data collections is essential to support many real-world applications ranging from healthcare to marketing [18]. Data mining is the process of extracting desirable knowledge or interesting patterns from existing databases for specific purposes. Many types of knowledge and technology have been proposed for data mining [38]. Among them, finding association rules or attribute implications is the most commonly seen.

Formulas of the form $A \Rightarrow B$ where A and B are collections of attributes have been studied for a long time in computer science and mathematics. In Formal Concept Analysis (FCA), formulas $A \Rightarrow B$ are called attribute implications. Attribute implications are interpreted in formal contexts, i.e. in data tables with binary attributes, and have the following meaning: Each object having all attributes from A has also all attributes from B (see e.g. [14, 34, 37]). In databases, formulas $A \Rightarrow B$ are called functional dependencies. Functional dependencies are interpreted in relations on relation schemes, i.e. in data tables with arbitrarily-valued attributes and have the following meaning: Any two objects which have the same values of attributes from A, have also the same values of attributes from B, see e.g. [5, 14, 47].

Generally, attribute implications are formulas of the form $A \Rightarrow B$, where A and B are collections of attributes, which describe dependencies between attributes [14].
As an example, assume a data set of requirements for eligible drivers and attribute implication of $\{Driver\} \rightarrow \{Driver's Licence, Car Insurance, Car\}$. This association rule indicates that to be able to drive, you have to have driver's licence, car insurance, and a car.

Formal Concept Analysis (FCA) was introduced in the early 1980s by Rudolf Wille as a mathematical theory [34, 39, 57]. Formal Concept Analysis is concerned with the formalisation of concepts and conceptual thinking and has been applied in many disciplines such as software engineering, machine learning, knowledge discovery, and ontology construc-

tion during the last 20-25 years. Informally, FCA studies how objects can be hierarchically grouped together with their common attributes [39]. Formal Concept Analysis (FCA) is a conceptual framework for structuring, analyzing, mining, and visualizing scientific data. From a formal context, which primarily describes binary relation between objects and attributes, FCA produces two outputs: first is the hierarchically ordered collection of clusters called formal concepts, and the second is the non-redundant basis of attribute dependencies called attribute implications. The goal of FCA is to discover the formal concepts – data dependencies in the form of attribute implications – and visualize them by a concept lattice [44].

To interpret a formal context using a table, rows and columns correspond to objects and attributes, respectively. The entries in the table can be 0 or 1, meaning whether an object has a specific property or not.

It might not always be obvious whether an object has a certain attribute or not. In this case, we have to use another concept known as fuzzy set. With more detail, in classical set theory, the membership of elements in a set is assessed in binary terms according to a bivalent condition — an element either belongs or does not belong to the set, i.e., the membership function of elements in the set is one or zero. By contrast, fuzzy set theory permits the gradual assessment of the membership of elements in a set. The membership function is valued in the real unit interval of [0, 1] [12].

The basic structure of FCA is the formal context which is a binary relation between a set of objects and a set of attribute. The formal context is based on the ordinary set, which elements have on the two values, 0 or 1 (Fig. 1.1) [68].

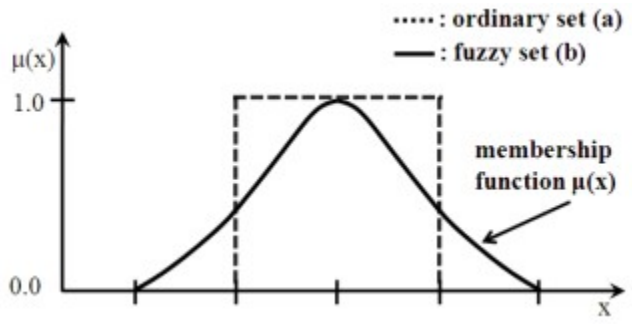


Figure 1.1: Ordinary set and fuzzy set.

Besides the ordinary set, fuzzy set theory permits uncertainty information that is directly represented by membership value in the range of [0, 1] (Fig. 1.1). The membership value which is taken through membership function indicates the grade of membership of set ele-

ments. If an element is mapped to the value 0, the element is not included in the fuzzy set, and 1 describes a fully included element [27, 68].
The truth degree by which an object has an attribute is usually taken from an appropriate lattice $\mathcal{L}$. In other words, this truth degree is the same as mentioned assessment of the membership for elements in a fuzzy set theory. A typical choice for $\mathcal{L}$ is the real unit interval [0,1] or some subset of thereof [12]. It means that $\mathcal{L}$ can take any value. So, the unit interval [0,1] and the Boolean values are two special cases of such a lattice $\mathcal{L}$.

Conceptual scaling is a way provided by FCA to manage a many-valued context, i.e., a context with $\mathcal{L}$ different from [0, 1], and transform it into a one-valued context. One-valued context is known as a Boolean context. In summary, conceptual scaling transforms any table using $\mathcal{L}$ values to a Boolean context. This transformation is based on certain rules.
More precisely, Table 1.1(a) shows a many-valued context which lists different water pipes having different attribute values. In order for FCA theory to be applied to a many-valued context, it needs to be unfolded into a one-valued context through conceptual scaling [34]. Table 1.1(b) shows the one-valued context for the many-valued context in Table1.1(a) after conceptual scaling [29].

Table 1.1: context transformation using conceptual scaling.

	what	how
pipeType1	foul	gravity
pipeType2	sludge	pressure

A many-valued context.

	foul	sludge	gravity	pressure
pipeType1	X		X	
pipeType2		X		X

(b) A one-value context after conceptual scaling

(a)

Generally, multiple approaches are proposed for applying FCA to fuzzy relations. First approach is transformation of the fuzzy context to a Boolean context and then apply regular (crisp) FCA on the derived relation(s). This can be done using conceptual scaling [32, 52]. So, such existing fuzzy approaches assume that the relationship between a given object and a given property is a matter of degree in a scale $\mathcal{L}$ (generally [0,1]). However, the extent to which "object o has property a" may be sometimes hard to assess precisely [26].
Therefore, several new scaling techniques have been proposed to solve the problem of determining the truth value from the scale $\mathcal{L}$. But there are still some problems with the proposed scaling methods.
In [26], using a sub-interval from the scale $\mathcal{L}$ rather than a precise value was proposed. Such formal contexts naturally lead to Interval-Valued Fuzzy Formal concepts (in short IVFF context). Also in [16], the representation of a fuzzy formal context using sub-interval

from the scale $\mathcal{L}$ was discussed.
We can find another scaling technique in [13] considering a static set of the fuzzy attribute to normalize the data. This is a new way of scaling, namely, scaling of general attributes to fuzzy attributes. The mentioned paper deals with scaling within the framework of formal concept analysis (FCA) of data with fuzzy attributes. In ordinary FCA, the input is a data table with yes/no attributes. Scaling is a process of transformation of data tables with general attributes, e.g. nominal, ordinal, etc., to data tables with yes/no attributes. This way, data tables with general attributes can be analyzed by means of FCA. After such a scaling, the data can be analyzed by means of FCA developed for data with fuzzy attributes.
The most important disadvantage of this proposed scaling method is the arbitrariness of boundaries of the fuzzy attribute set. Therefore, this technique is very sensitive because of the user's selection of scale attributes and results are in a large difference in the generated concepts lattice with a small change in scale attribute [13].
Most of the existing works published so far have contained unfavorable consequences based on the conversion of the data from a multi-valued (fuzzy) context to a Boolean context before applying Formal Concept Analysis. On the other hand, making a decision on the boundaries between scale attributes is biased by the user intention, since it is impossible to find a unique rational thought behind selection of attribute boundaries.

Second approach is working with the unit interval [0, 1] and specific *t*-(co)norm (*left continuous t-norms*) operation to define fuzzy implications underlying the fuzzy FCA notions. The reason why several papers used left-continuous t-norm is that this guarantees that the t-norm has a right adjoint. In these papers, a complete lattice can be formed by fuzzy formal concepts [41, 46].

Next approach belongs to Radim Bělohlávek and others [12, 15, 51] that considered *complete residuated lattices* as the domain of potential truth values. This is based on considering the table entries as truth degrees in fuzzy logic and proceeding by the basic setting of FCA, just replacing classical logic with fuzzy logic in FCA". It means that they work with fuzzy formal concept analysis (FFCA) and fuzzy concept lattices are generated.
Bělohlávek et. al. (2005, and then a followed up in 2011) used the same approach as we do but they did this for *concrete $\mathcal{L}$-fuzzy* relations only, i.e., relations with a characteristic function $A \times B \to \mathcal{L}$ where A and B are sets and $\mathcal{L}$ is a complete Heyting algebra. In particular for *concrete $\mathcal{L}$-fuzzy* relations both theories are the same. However, we do this also using the abstract theory of *arrow / fuzzy categories* [1].
In other words, Bělohlávek and others used *regular first-order logic* to formulate every-

thing. It means that all definitions of operations, relations, and sets are component-wise. We do this in an algebraic manner, i.e. we treat relations as elements of algebraic structures such as arrow and fuzzy categories. Also, we define the operations, etc., using the operations on relations.

A significant benefit of our approach is that the properties and theorems are not referring to the potential involved elements of the sets. Linear algebra behaves in a similar way while expresses properties of linear maps. It uses the operations on maps instead of the component(element)-wise reasoning. An equational first-order theory is the result of this in which all theorems are defined in an algebraic manner. Such a simple representation can lead to an attempt to further develop the theory and its use in applications. e.g. RelView [1], so that a simple relation-algebraic term for concrete applications can be calculated to define the order relation of a concept lattice. Since equational theories are convenient to be used in (semi) automatic theorem provers, the implementation of the theory developed in this thesis is very straightforward using any system of type theory such as Coq [2]. The consequence is the presence of the fuzzy concept analysis (FCA) in libraries of proofs. So, the computing components generated by verified programs can be available in several programming languages using FCA implementation. Meanwhile, our algebraic framework is able to cover fuzzy concept analysis as well as regular (crisp) concept analysis. As another major advantage of our approach, it is worth mentioning that arrow/fuzzy categories contain some models. Such models are not the same as categories of concrete fuzzy relations, i.e., our approach is slightly general [9, 12, 15, 51] and can cover both regular and $\mathcal{L}$-Fuzzy FCA, simultaneously.

1.1 Main Contributions

1. Develop a $\mathcal{L}$-Fuzzy Concept Analysis as a generalization of a regular Concept Analysis in a way that it should work with the $\mathcal{L}$-Fuzzy data directly (not transforming the $\mathcal{L}$ Fuzzy formal context to a crisp one). This technique is performed without using data pre-processing (transferring), and then FCA is applied directly on the fuzzy contexts (relations). Data pre-processing means transformation of the fuzzy context to a Boolean context and then apply regular (crisp) FCA on the derived relation(s). So, we work on original data without any preliminary transformation step.

2. Generate a framework of $\mathcal{L}$-fuzzy formal concept analysis with the aim of addressing

[1] `https:\\www.rpe.informatik.uni-kiel.de\en\research\relview` (homepage of RelView).
[2] `https:\\coq.inria.fr` (homepage of Coq).

a multi-valued context regardless of the data preprocessing stage using a relational approach. These relational algebraic formulations are used to generate the concepts and attribute implications from a given context including a Boolean as well as a multi-valued contexts. It means that this algebraic framework can cover both regular and $\mathcal{L}$-Fuzzy FCA simultaneously.

3. Manage the proposed ambiguity found in the object classification process and receive the final output (attribute implications) extracted from the original input data with minimal uncertainty.

4. Implement the approach by Java programming language.

1.2 Thesis Structure

The structure of this thesis has been outlined in six following chapters:

- **Chapter 1** provides a general introduction and overview of the reason behind the selection of the subject of the relational approach to $\mathcal{L}$-FCA as well as the way to implement it.

- **Chapter 2** covers the relevant mathematical preliminaries including some illustrative examples from the basic definitions of $\mathcal{L}$-fuzzy relations to various categorical approaches. The concepts in this chapter are provided for a more accurate understanding of the topic presented in this dissertation.
  The chapter is organized into four main sections, introducing lattices, relations, $\mathcal{L}$-fuzzy relations, and categories of relations.

- **Chapter 3** mainly includes the definitions of Formal Concept Analysis (FCA) [25] and Fuzzy Formal Concept Analysis (FFCA) [46]. To rephrase it, the background theories of the two concept analysis are described in detail using several related examples in this chapter. Introducing relational algebraic formulas for formulating formal concept analysis is another focus of this section.

- **Chapter 4** concentrates on concept lattices in allegories and attribute implications. The chapter includes the main contribution of this thesis to extract attribute implications and prepare them for implementation in the next and final phase.

- **Chapter 5** proposes concrete implementation for studies presented in chapters 3 and 4 of this dissertation. In order to enhance understanding of implementation phase,

two main sections are included in this chapter; basic algebraic formulas in the library along with their applications and the final framework.

- **Chapter 6** expresses the concluding remarks of this thesis study and some potential avenues for future works.

Chapter 2

Relation-Algebraic Preliminaries

This chapter contains some basic mathematical background used in this thesis. A full understanding of these concepts and operations is provided using some relevant concrete examples. All major notations of lattice and set theories are essential to the understanding of our research work.

2.1 Lattices

A very natural concept is a partially ordered set (or poset). Elements of such a set may be related to each other by notion of ”being smaller or equal”. Formally, poset, linear, and non-linear posets are defined as below [61]:

Definition 2.1.1. A *poset* is a set P with a binary relation $\leq$ on it so that

(i) ***reflexive*** $x \leq x$ for all $x \in P$,

(ii) ***transitive*** if $x \leq y$ and $y \leq z$, then $x \leq z$ for all $x, y, z \in P$,

(iii) ***antisymmetric*** if $x \leq y$ and $y \leq x$, then x = y for all $x, y \in P$.

Definition 2.1.2. A poset P is called *linear* iff $x \leq y$ or $y \leq x$ holds for all $x, y \in P$.

Example 2.1.1.

1. **Linear Poset** the set of real numbers $\mathbb{R}$ with the usual ordering is a linear poset. Obviously, the unit interval [0, 1] of the real numbers is also a linear poset.

2. **Non-linear Poset** the power set P(A) of a set A with more than one element together with set-inclusion $\subseteq$ is a example of a non-linear poset.

   The Hasse diagram of the set of all subsets of a three-element set $\{x, y, z\}$ is shown in Fig. 2.1. Sets connected by an upward path, like $\emptyset$ and $\{x, y\}$, are comparable, while e.g. $\{x\}$ and $\{y\}$ are not.

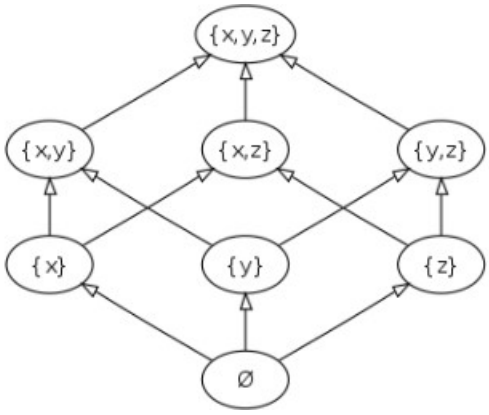


Figure 2.1: Hasse diagram of the powerset of $\{x, y, z\}$.

Simply put, when posets are finite, it is possible to represent P as a triangular matrix or a Hasse diagram. A fuzzy Hasse diagram is a valued, oriented graph whose nodes are the elements of the poset. The link $x \to y$ in a Hasse diagram exists iff $\mu_p(x, y) > 0$, i.e. each link is valued by $\mu_p(x, y) > 0$ (μ shows existence of an element (x, y) in the poset). Owing to perfect antisymmetry and transitivity, the graph has no cycle. An example (Zadeh, 1971) is provided in Fig. 2.2 [19].

In other words, the Hasse diagram represents a partial order between the symplectic leaves, defined by inclusions in their closures. For any two given leaves which can be compared in this partial order, we have a transverse slice which describes how the smaller leaf looks as a simple singularity inside the closure of the bigger leaf similar to inclusions [55].

Definition 2.1.3. A preorder $\subseteq$ is said to be a partial order if it is anti-symmetric also. If $\subseteq$ is a partial order on X then the pair (X,$\subseteq$) is said to be a poset. Given a set $A \subseteq X$, x is said to be an *upper bound* of A (*lower bound*) if for every $y \in A$, we have that $y \subseteq x$ ($x \subseteq y$ respectively).

An element z is said to be a *least upper bound* of A (*greatest lower bound* of A) if z is an upper bound (lower bound respectively) and for every upper bound (lower bound respectively) x of A, we have that $x \subseteq z$ ($z \subseteq x$ respectively) [21]. [1]

[1]The least upper bound and the greatest lower bound may not exist. But, if they exist, they are unique.

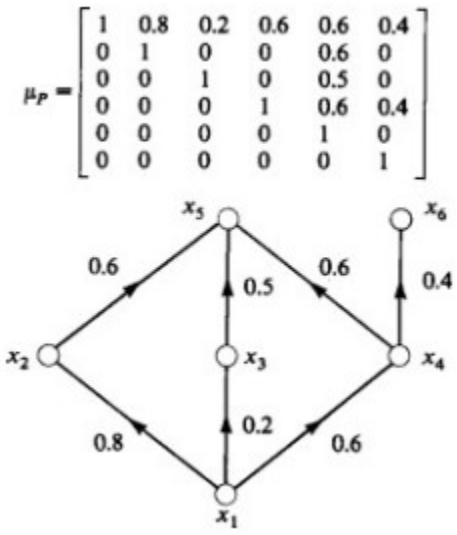


Figure 2.2: A fuzzy Hasse diagram.

Example 2.1.2. As illustrated in Fig. 2.3, if x and y are elements of a partial order, an upper bound for x and y is simply an element u such that $x \leq u$ and $y \leq u$. u is the least upper bound of x and y if u is ≤ all upper bounds of x and y.
Also, the set of upper bounds for n and g is $\{r, s, t, u, d, a\}$. Obviously, there is a way to get to each of them from both n and g by travelling upwards in the diagram. On the other hand, i is not an upper bound for n and g because: $n \leq i$, but $g \not\leq i$.
Since $r \leq r$, $r \leq s$, $r \leq t$, $r \leq u$, $r \leq d$, and $r \leq a$, the lowest member of that set is r. That makes r the least upper bound of n and g.
The elements above j are b, c, e, m, l, a, k, d, i, t, u, and the elements above g are r, d, a, u, s, t. The elements that are above both j and g are a, d, u, t. So, these four elements are the upper bounds of j and g. Here, however, there is no least upper bound: $d \leq d$, $d \leq a$, and $d \leq u$, but $d \not\leq t$ (none of a, u, and t is $\leq d$).

Definition 2.1.4. A *lattice* L is a poset such that for all elements $a, b \in L$ there exists the *supremum* or *join* $a \vee b$ (*the least upper bound* $a \vee b$ of the set $\{a, b\}$ on P) and the *infimum* or *meet* $a \wedge b$ (the *greatest lower bound* $a \wedge b$ of the set $\{a, b\}$ on P). Note that, for an arbitrary poset P , $a \vee b$ and $a \wedge b$ may not exist. [3, 34]

Definition 2.1.5. A *complete lattice* is a lattice in which any subset A has a least upper bound and a greatest lower bound [3, 48, 61].

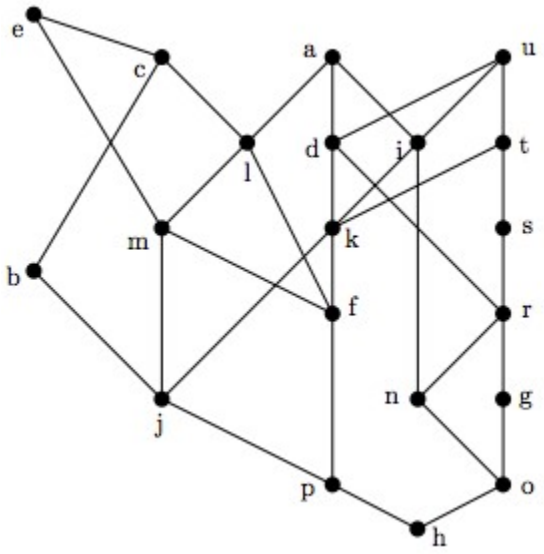


Figure 2.3: The greatest lower bound and the least upper bound using Hasse diagram of a partial order.

Definition 2.1.6. We can define the notion of a lattice also algebraically. The binary operations $\wedge$ and $\vee$ of the corresponding lattice fulfills the following properties [61]

(i) ***idempotent*** $x \wedge x = \mathrm{x}$, $x \vee x = \mathrm{x}$,

(ii) ***commutative*** $x \wedge y = y \wedge x$, $x \vee y = y \vee x$,

(iii) ***associative*** $x\wedge (y \wedge z) = (x \wedge y) \wedge \mathrm{z}$, $\mathrm{x} \vee (y \vee z) = (x \vee y) \vee \mathrm{z}$,

(iv) ***consistent*** $x \wedge y = \mathrm{x} \Leftrightarrow x \leq y$, $x \vee y = \mathrm{y} \Leftrightarrow x \leq y$,

(v) ***monotone*** $\forall\, y \leq z : x \wedge y \leq x \wedge z$, $x \vee y \leq x \vee z$.

Definition 2.1.7. A lattice X is called *bounded* if there exist two elements $0_X, 1_X \in X$ such that [48]

$$0 \wedge x = 0, \quad 0 \vee x = \mathrm{x}, \quad 1 \wedge x = \mathrm{x}, \quad 1 \vee x = 1.$$

In basic terms, a bounded lattice is a lattice with least element 0 and greatest element 1. Given the definition of a bounded lattice, a complemented lattice can be defined as below.

Definition 2.1.8. A lattice X is called *complemented* if it is bounded and for every $x \in X$ there exists $y \in X$ such that [48]

$$x \vee y = 1, \quad x \wedge y = 0.$$

More detailed explanation of a complemented lattice can be found in [36].

Definition 2.1.9. A lattice X is called *distributive* [48] if for every $x, y, z \in X$, the meet ($\wedge$) and the join ($\vee$) operations are distributed over each other,

$$\mathrm{x} \vee (y \wedge z) = (x \vee y) \wedge (x \vee z), \qquad \mathrm{x} \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

Example 2.1.3. Fig. 2.4 is shown a a non-distributive lattice, since[2]:

$$\mathrm{x} \wedge (y \vee z) = x \wedge 1 = x \neq 0$$
On the other hand we have:
$$(x \wedge y) \vee (x \wedge z) = 0 \vee 0 = 0$$

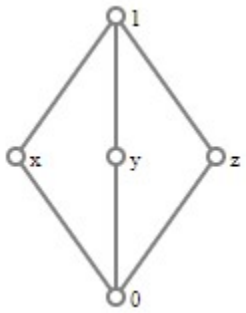


Figure 2.4: A non-distributive diamond lattice.

2.1.1 Complete Heyting Algebra

The language of lattices is $L_{lat} = \{ 0, 1, \vee, \wedge \}$, where 0 stands for the smallest element, 1 for the greatest one, $\vee$ and $\wedge$ for the join and meet operations respectively. Of course the order of the lattice is definable by $b \leq a$ if and only if $a \vee b = \mathrm{a}$. The language of *Heyting algebras* is $L_{HA} = L_{lat} \cup \{ \rightarrow \}$ where the new binary operation symbol (implication operation) is interpreted as [24]

$$b \rightarrow a = \max \{ \mathrm{c} : c \wedge b \leq a \}.$$

[2] https://en.wikipedia.org/wiki/Distributive_lattice

An example of this implication is provided later in this thesis in Example. 2.3.3.
This binary operation can also be referred as *relative pseudo-complement*. A Heyting algebra is said to be *complete* if it is a complete lattice. Another way of defining Heyting Algebra and relative pseudo-complement can be found in the following definition:

Definition 2.1.10. Let L be a bounded lattice and $x, y \in L$. A *relative pseudo-complement* $x \to y$ of x in y is an element so that [61]

$$u \leq x \to y \iff x \wedge u \leq y$$

for all $u \in L$. A lattice in which for every pair of elements the relative pseudo-complement exist is called a *Heyting algebra*. In the following we will use the abbreviation $x \leftrightarrow y$ for $(x \to y) \wedge (y \to x)$.

We used relative pseudo-complement later in this thesis in Example. 2.3.3 to calculate the implementation matrix, left residual matrix, etc. among fuzzy relations.

2.2 Binary Relations

For the sake of obtaining a clear definition of binary relation R, the concept of Cartesian product of two sets is a major one. Given two sets of objects, the operation of Cartesian product forms ordered pairs in mathematics as below:

Definition 2.2.1. The *Cartesian product* of two sets A and B is the set of all pairs (x,y) with $x \in A$ and $y \in B$, and is denoted by $A \times B$ [61].

$$A \times B = \{ (x, y) : x \in A \text{ and } y \in B \}$$

Example 2.2.1. Considering some bodies of water[3] including the objects (set of O) and their attributes (set of A), Cartesian product is derived as:

Objects = { Lagoon, Lake, Maar, Pond, Pool, Sea, Tarn }
Attributes = { Natural, Stagnant, Constant }
$O \times A$ = { (Lagoon, Natural), (Lagoon, Stagnant), (Lagoon, Constant), (Lake, Natural), (Lake, Stagnant), (Lake, Constant), (Maar, Natural), (Maar, Stagnant), (Maar, Constant), (Pond, Natural), (Pond, Stagnant), (Pond, Constant), (Pool, Natural), (Pool, Stagnant), (Pool, Constant), (Sea, Natural), (Sea, Stagnant), (Sea, Constant), (Tarn, Natural), (Tarn, Stagnant), (Tarn, Constant) }

[3] https://en.wikipedia.org/wiki/Formal_concept_analysis

A relation R between two arbitrary sets A and B is a set of pairs of elements from A and B, i.e., $R \subseteq A \times B$, and can be denoted by R: $A \rightarrow B$. This implies that binary relation R between two sets is a subset of the Cartesian product of the two sets. A more accurate definition of relation R is as follows:

Definition 2.2.2. A *binary relation* R between two sets A and B is an element of Power set $A \times B$. A is called the source and B the target of R.
If A = B, the relation R is also called an *endorelation* or *homogeneous*. To indicate that a binary relation R has source A and target B we usually write $R \subseteq A \times B$ [10, 61].

Using matrix visualization, a better understanding of finite binary relations between two finite sets of objects and attributes can be obtained. The dimensions of the matrix are determined by the number of elements in the object and attribute sets, i.e., the rows and columns of the matrix represent the objects and attributes, respectively. In order to obtain a matrix representation, we assume that the elements of the source A and target B are ordered, i.e., we assume that $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$. In such a case, if there is a relationship between an object and a particular attribute, the associated element in the matrix is 1, otherwise it is denoted by 0. If we denote the entry in the i-th row and j-th column of the matrix of the relation R by $M_{i,j}$, then we have:

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

Example 2.2.2. Considering bodies of water example again, for two sets of objects and attributes the associated binary relation in the form of a binary matrix is shown as follows:

Objects = { Canal, Channel, Lagoon, Lake, Reservoir, Torrent }
Attributes = { Running, Natural, Stagnant, Constant, Maritime }

$$\boldsymbol{R} = \begin{array}{c} \\ Canal \\ Channel \\ Lagoon \\ Lake \\ Reservoir \\ Torrent \end{array} \begin{array}{c} \begin{array}{ccccc} Running & Natural & Stagnant & Constant & Maritime \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

2.2.1 Basic Operations and Properties on Binary Relations

Some primary operations on two relations Q and R are required to be defined. The next definition provides some of these operations. As previously mentioned in the prior section, $P : X \to Y$ is a subset of the Cartesian product of X and Y, i.e., $P \subseteq X \times Y$.
In terms of matrices, the basic operations can be generated using the Boolean operations AND($\wedge$), OR($\vee$), and NOT($\neg$) on the corresponding entries in the matrices. An extended version of the basic relations is used in the following sections on $\mathcal{L}$-fuzzy relations.

Definition 2.2.3. Let Q, R : $X \to Y$ be the same relations. Then we may introduce several operations among these two relations as follows:

1. ***Union Relation*** $Q \sqcup R := \{(x,y) \in X \times Y \mid (x,y) \in Q \vee (x,y) \in R\}$
2. ***Intersection Relation*** $Q \sqcap R := \{(x,y) \in X \times Y \mid (x,y) \in Q \wedge (x,y) \in R\}$
3. ***Converse Relation*** $Q^{\smile} := \{(y,x) \in Y \times X \mid (x,y) \in Q\}$
4. ***Complement Relation*** $\overline{Q} := \{(x,y) \in X \times Y \mid (x, y) \notin Q\}$
5. ***Implication Relation*** $Q \to R := \{(x,y) \in X \times Y \mid (x,y) \notin Q \vee (x,y) \in R\}$
6. ***Identity Relation*** $\mathbb{I}_X := \{(x,x) \in X \times X \mid x \in X\}$
7. ***Empty relation*** (Least elements) $\perp\!\!\!\perp_{XY} = \emptyset$ [4]
8. ***Universal Relation*** (Greatest elements) $\top\!\!\!\top_{XY} = X \times Y$ [5]

We will denote the fact that a relation Q is included in another relation R, i.e., a subset of, by $Q \sqsubseteq R$.

Example 2.2.3. Taking into consideration of two relations Q and R retrieved from some bodies of water examples other than the previous ones, a better visualization of basic operations can be obtained.

$$\boldsymbol{Q} = \begin{array}{c} \\ Pool \\ Reservoir \\ River \\ Rivulet \end{array} \begin{array}{c} \begin{array}{cccc} Temporary & Running & Natural & Stagnant \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

[4] The set with no elements is called the empty set, and is denoted by $\emptyset$ [61].

[5] This relation is also called the top relation, which is the largest relation that connects all the elements of X to the elements of Y.

$$\boldsymbol{R} = \begin{array}{c} \\ \mathit{Runnel} \\ \mathit{Sea} \\ \mathit{Stream} \\ \mathit{Tarn} \end{array} \begin{array}{c} \begin{array}{cccc} \mathit{Temporary} & \mathit{Running} & \mathit{Natural} & \mathit{Stagnant} \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

1. ***Union Relation***:

$$Q \sqcup R = \begin{array}{c} \\ \mathit{Pool} \\ \mathit{Reservoir} \\ \mathit{River} \\ \mathit{Rivulet} \end{array} \begin{array}{c} \begin{array}{cccc} \mathit{Temporary} & \mathit{Running} & \mathit{Natural} & \mathit{Stagnant} \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

2. ***Intersection Relation***:

$$Q \sqcap R = \begin{array}{c} \\ \mathit{Pool} \\ \mathit{Reservoir} \\ \mathit{River} \\ \mathit{Rivulet} \end{array} \begin{array}{c} \begin{array}{cccc} \mathit{Temporary} & \mathit{Running} & \mathit{Natural} & \mathit{Stagnant} \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

3. ***Converse Relation (Transpose)***:

$$Q^{\smile} = \begin{array}{c} \\ \mathit{Temporary} \\ \mathit{Running} \\ \mathit{Natural} \\ \mathit{Stagnant} \end{array} \begin{array}{c} \begin{array}{cccc} \mathit{Pool} & \mathit{Reservoir} & \mathit{River} & \mathit{Rivulet} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

4. ***Complement Relation***:

$$\overline{Q} = \begin{array}{c} \\ Pool \\ Reservoir \\ River \\ Rivulet \end{array} \begin{array}{c} \begin{array}{cccc} Temporary & Running & Natural & Stagnant \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

5. ***Implication Relation***:

$$Q \to R = \begin{array}{c} \\ Pool \\ Reservoir \\ River \\ Rivulet \end{array} \begin{array}{c} \begin{array}{cccc} Temporary & Running & Natural & Stagnant \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

2.2.2 Relational Operations and Properties

This section consists of some additional operations on relations. In the previous section, we started the relation definitions by the concrete boolean-valued relations, i.e., for relations R between two sets X and Y. Formally such a relation is a subset of the set X × Y of pairs from X and Y, or, alternatively, a function from X × Y to {True, False}. For the second version R(x,y) for x ∈ X and y ∈ Y is either True or False saying that either x is related to y in the relation R or not.
Using this, we get the following constructions for the relational operations. The mentioned additional relations include relational converse, composition, ϵ (epsilon) relation, right and left residuals, and symmetric quotient. Also at the end, using Q and R relations used in example 2.2.3, relational operations and properties are clarified by a few examples.

The operation of converse is exchanging the row and column. This operation is *monotone*. Just like the negation operator, applying the operation twice results in the original relation itself:

$$(R^{\smile})^{\smile} = \mathrm{R}.$$

Apart from the set theoretic operations, we consider one further operation on binary relations as follows.

Definition 2.2.4. Let P be a relation between A and B and R between B and C. Then we define *composition* between P and R as below [61]

$$P ; R := \{ (x, z) \mid \exists\, y \in B \quad :(x, y) \in P \wedge (y, z) \in R \}.$$

Due to the definition above, a composition $P \circ R$ has to be read from the left to the right, i.e., first P, and then R. We usually write R(x, y) instead of (x,y) ∈ R.
Notice that composition is associative, i.e., for all relation $P : A \to B$, $Q : B \to C$ and $R : C \to D$ we have (P ; Q) ; R = P ; (Q ; R) [61].

Example 2.2.4. Considering Q and R relations defined in example 2.2.3, the composition of Q and R returns the following relation.

$$\boldsymbol{Q ; R} = \begin{array}{c} \\ Pool \\ Reservoir \\ River \\ Rivulet \end{array} \begin{array}{c} \begin{array}{cccc} Temporary & Running & Natural & Stagnant \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

The only necessary condition for combining two matrices is that the number of columns of the first matrix should be equal to the number of rows of the second matrix.

Some important classes of relations and properties are derived by mappings [60].

Lemma 2.2.1. Let Q: A → B, be a relation. Then we call:

(i) Q is univalent iff $Q^\smile ; Q \sqsubseteq \mathbb{I}_B$.

(ii) Q is total iff $\mathbb{I}_A \sqsubseteq Q ; Q^\smile$ or equivalently iff $Q ; \pi_{BC} = \pi_{AC}$ for all objects C.

(iii) Q is a map iff Q is univalent and total.

(iv) Q is injective iff $Q^\smile$ is univalent.

(v) Q is surjective iff $Q^\smile$ is total.

(vi) Q is an isomorphism iff Q and $Q^\smile$ are both mappings.

Notice that if Q is an isomorphism $Q^\smile ; Q = \mathbb{I}_B$ and $Q ; Q^\smile = \mathbb{I}_A$ hold.
Proofs of these properties including some more others, can be found in [54, 60].

2.2.2.1 Projection Operations

The following definition ascertains relational product which is a type of relational construction given by the Cartesian products of sets. The mentioned relational products are known as projection operations including Pi (π) and Rho (ρ). We will refer to projection operations in the Section. 2.4.7 (Relational Product in Dedekind categories) further in this thesis.

Definition 2.2.5. Let A = { a, b } and B = { c }. Then, two relations π and ρ are defined by ((a,b), c)$\in \pi$ iff c = a and ((a,b), c) $\in \rho$ iff b = c, respectively.

Example 2.2.5. Consider two sets A = { a_1, a_2 } and B = { b_1, b_2 }. π and ρ can be generated as below:

$$\pi = \begin{array}{c} \\ \langle a_1,b_1\rangle \\ \langle a_1,b_2\rangle \\ \langle a_2,b_1\rangle \\ \langle a_2,b_2\rangle \end{array} \begin{array}{c} \begin{array}{cc} a_1 & a_2 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \end{array}$$

$$\rho = \begin{array}{c} \\ \langle a_1,b_1\rangle \\ \langle a_1,b_2\rangle \\ \langle a_2,b_1\rangle \\ \langle a_2,b_2\rangle \end{array} \begin{array}{c} \begin{array}{cc} b_1 & b_2 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \end{array}$$

In this example, column-values in matrices correspond to the value of c in the above definition for *Pi* and *Rho*.

The properties represented below on projection operations can be found in [61].

Lemma 2.2.2.

$$\pi^\smile ; \pi \sqsubseteq \mathbb{I}_A, \quad \rho^\smile ; \rho \sqsubseteq \mathbb{I}_B, \quad \pi^\smile ; \rho = \top_{AB}, \quad \pi^\smile ; \pi \sqcap \rho^\smile ; \rho = \mathbb{I}_{A\times B}$$

These properties characterize a product and will be used as the abstract definition later in this thesis.

2.2.3 Combined Operations and Properties

This section describes some composite relations derived by some basic boolean operations such as implication. Our focus in this section is on residuals (Right and Left), and symmetric quotient operations, which are among the most principal ones in this research. Using two various techniques, the definitions of right and left residuals are released as:

Definition 2.2.6.

1. ***Left Residual***:
   Let Q: X → Y and R: X → Z, then *left Residual* is defined by **Q** \ **R**: Y → Z with:

$$\mathbf{Q} \setminus \mathbf{R} = \{(y, z) \mid \forall\, x\text{: } (x, y) \in Q \rightarrow (x, z) \in R\}.$$

2. ***Right Residual***:
   Let Q: X → Y and R: Z → Y, then *right Residual* is defined by **Q / R**: X → Z with:

$$\mathbf{Q} / \mathbf{R} = \{(x, z) \mid \forall\, y\text{: } (z, y) \in R \rightarrow (x, y) \in Q\}.$$

Example 2.2.6.

$$\boldsymbol{Q} = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{cccc} y_0 & y_1 & y_2 & y_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \end{array}, \quad \boldsymbol{R} = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{cccc} z_0 & z_1 & z_2 & z_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Considering the above two relations we have the matrix representation of the ***left residual*** as:

$$\boldsymbol{Q \backslash R} = \begin{array}{c} \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{array} \begin{array}{c} \begin{array}{cccc} z_0 & z_1 & z_2 & z_3 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

In matrix representation of left residual for boolean relations, we get a value of 1 for an

element (y, z) in the left residual matrix iff the column of y in Q is included in the column of z in R. *Included* means every entry 1 in the first column (in Q) must have a 1 in the second column (in R) at the same place.

Example 2.2.7.

$$\boldsymbol{Q} = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} y_0 & y_1 & y_2 & y_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \end{array}, \quad \boldsymbol{R} = \begin{array}{c} \\ z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \end{array} \begin{array}{c} \begin{array}{cccc} y_0 & y_1 & y_2 & y_3 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Considering the above two relations we have the matrix representation of the ***right residual*** as:

$$\boldsymbol{Q/R} = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{ccccc} z_0 & z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

In matrix representation of right residual for boolean relations, the only difference compared to left residual is that we consider rows instead of columns in two relations, and then we have to check "*Including*" between the two rows.

More details regarding definitions of residuals, the proofs, and their properties can be found in [53, 54].

Our next focus is on symmetric Quotient among combined operations. This operation is indicated by abbreviation of *syq* in the main sources of science. There exist the same two techniques for Symmetric Quotient just like residuals:

Definition 2.2.7. The symmetric quotient syq(Q, R): $Y \to Z$ of two relations Q: $X \to Y$ and R: $X \to Z$ is defined by:

$$\boldsymbol{syq(Q, R)} = \{\ (y, z) \in Y \times Z \quad | \quad \forall x \in X\colon \ \ (x, y) \in Q \longleftrightarrow (x, z) \in R\ \}.$$

Example 2.2.8.

$$\boldsymbol{Q} = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{cccc} y_0 & y_1 & y_2 & y_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array}, \quad \boldsymbol{R} = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccc} z_0 & z_1 & z_2 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$\boldsymbol{syq(Q,\ R)} = \begin{array}{c} \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{array} \begin{array}{c} \begin{array}{ccc} z_0 & z_1 & z_2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

Definition 2.2.8. The relation ϵ: A $\rightarrow \mathcal{P}$(A) where $\mathcal{P}$(A) is the power set[6] of A, is defined by (x, M) $\in \epsilon$ iff x $\in$ M.

Lemma 2.2.3. Using simple set A = { a_1, a_2, a_3 }, related ϵ relation has 8 members represented as below:

$$\{ a_1, a_2, a_3 \} \rightarrow \{ \{ \}, \{ a_1 \}, \{ a_2 \}, \{ a_3 \}, \{ a_1, a_2 \}, \{ a_1, a_3 \}, \{ a_2, a_3 \} \{ a_1, a_2, a_3 \} \}$$

$$\boldsymbol{\epsilon} = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array} \begin{array}{c} \begin{array}{cccccccc} \{\} & \{a_1\} & \{a_2\} & \{a_3\} & \{a_1,a_2\} & \{a_1,a_3\} & \{a_2,a_3\} & \{a_1,a_2,a_3\} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

The proof is an easy exercise. We will add this as a definition in the section of abstract allegories 2.4.2.
In the next definition, some properties of the ϵ relation on boolean values are discussed [64].

Definition 2.2.9.
Consider the following two properties:

- syq(ϵ, ϵ) = $\mathbb{I}$

[6] In mathematics, the power set of a set A is the set of all subsets of A, including the empty set and A itself

- syq(R, ϵ) is *total* for every R.

The first assertion means that ϵ is *extensional*, i.e., two elements of $\mathcal{P}$(A) are equal iff they contain the same elements, or each column of ϵ (the elements of the set corresponding to that column) are different. The second property says that every relation R can be transformed into a function syq(R, ϵ) so that this function maps a y to the set {x | (x, y) ∈ R}.
As mentioned in Lemma. 2.2.1 (ii), a relation Q: A ⟶ B is *total* iff $\mathbb{I}_A \sqsubseteq$ Q ; $Q^\smile$ (or equivalently Q ; $\pi_{BC} = \pi_{AC}$ for all objects C). In Boolean matrices this means that there is at least one element 1 in each row.

2.3 $\mathcal{L}$-Fuzzy Relations

Among relations two types are of interest in this research. First, a relation R is called 0-1 *crisp* iff R(x, y) = 0 or R(x, y) = 1 holds for all x and y. Crisp relations may be identified with classical (binary) relations [62]. The second type of relations that are of interest are *fuzzy* relations. In this types of relations, there is a weighing in the relationship between two objects with a degree.
To get more clarification, in the basic setting, attributes are assumed to be binary, i.e. table entries are 1 or 0 according to whether an attribute applies to an object or not. If the attributes under consideration are fuzzy (like "cheap", "expensive"), each table entry contains a truth degree to which an attribute applies to an object. The degrees can be taken from some appropriate scale containing 0 (does not apply at all) and 1 (fully applies) as bounds. The most popular choice is some subinterval of [0, 1], but in general, degrees need not to be numbers [31].

In mathematics, especially in order theory, a *complete Heyting algebra* is a Heyting algebra that is complete as a *lattice*[7]. We consider $\mathcal{L}$ to be always a *complete Heyting algebra* in this section. An $\mathcal{L}$-fuzzy relation Q: A → B is a function Q: A × B → $\mathcal{L}$. Regular relations are *Boolean-valued fuzzy relations* since a subset of the Cartesian product A × B can also be represented by the characteristic function f_Q: A × B → $\mathbb{B}$ so that $\mathbb{B}$ contains Boolean values.

[7] A *complete lattice* is a partially ordered set in which all subsets have both a *supremum* (join) and an *infimum* (meet). A lattice which satisfies at least one of these properties is known as a *conditionally complete lattice*. Specifically, every *non-empty finite lattice* is complete.

Example 2.3.1. ***$\mathcal{L}$-Fuzzy Relations***: As an example, consider $X = \{0, \ldots, 10\}$ as source and a similar finite set $Y = \{0, \ldots, 10\}$ as target. For an element $u \in \mathcal{L}$, we define the corresponding fuzzy relation (R) by:

$$R(x, y) := \begin{cases} u & \textit{iff} \quad y \geq x, \\ 0 & \textit{otherwise}. \end{cases}$$

Similar to the matrix representation for regular relation we can visualize an $\mathcal{L}$-fuzzy relation by an $\mathcal{L}$-valued matrix. In the example above we obtain:

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{array} \begin{array}{c} \begin{array}{ccccccccccc} y_0 & y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 & y_{10} \end{array} \\ \left(\begin{array}{ccccccccccc} u & u & u & u & u & u & u & u & u & u & u \\ 0 & u & u & u & u & u & u & u & u & u & u \\ 0 & 0 & u & u & u & u & u & u & u & u & u \\ 0 & 0 & 0 & u & u & u & u & u & u & u & u \\ 0 & 0 & 0 & 0 & u & u & u & u & u & u & u \\ 0 & 0 & 0 & 0 & 0 & u & u & u & u & u & u \\ 0 & 0 & 0 & 0 & 0 & 0 & u & u & u & u & u \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & u & u & u & u \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u & u & u \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u & u \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u \end{array} \right) \end{array}$$

As an example, $\mathcal{L} = \{0.2, 0.4, 0.6, 0.8, 1\}$ which is in the range of (0, 1], or any other arbitrary decimal number within this scale.

Example 2.3.2. ***$\mathcal{L}$-Fuzzy Relations***: We are going to consider an illustrative example given in [11, 8]. Our frame is composed of the lattice $\mathcal{L} = \{0, \frac{1}{2}, 1\}$.
In Table 2.1 , the set of objects contains nine elements (Mercury (Me), Venus (V), Earth (E), Mars (Ma), Jupiter (J), Saturn (S), Uranus (U), Neptune (N), Pluto (P)), and the set of attributes contains the small size, large size, far from the Sun, and near the Sun.

Table 2.1: Fuzzy relation R between the objects and the attributes.

	Me	V	E	Ma	J	S	U	N	P
small size	1	1	1	1	0	0	$\frac{1}{2}$	$\frac{1}{2}$	1
large size	0	0	0	0	1	1	$\frac{1}{2}$	$\frac{1}{2}$	0
far from the Sun	0	0	0	$\frac{1}{2}$	1	1	1	1	1
near the Sun	1	1	1	1	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0

2.3.1 Basic Operations and Properties on $\mathcal{L}$-Fuzzy Relations

This section describes extensions or generalizations of basic operations defined for regular(crisp) binary relations to the $\mathcal{L}$-Fuzzy Relations, i.e, for $\mathcal{L} = \mathbb{B}$. The mentioned operations are discussed in the Section. 2.2.1.

Definition 2.3.1. Let Q, R: X $\to$ Y and S: Y $\to$ Z denote some $\mathcal{L}$-Fuzzy relations. Then we may introduce several primary operations as follows [61]:

1. ***Union*** $(Q \sqcup R)(x, y) = Q(x, y) \vee R(x, y)$,
2. ***Intersection*** $(Q \sqcap R)(x, y) = Q(x, y) \wedge R(x, y)$,
3. ***Conversion*** [8] $Q^{\smile}(x, y) := Q(y, x)$,
4. ***Composition***
$$(Q; S)(x, z) = \bigvee_{y \in B} Q(x, y) \wedge S(y, z)$$
5. ***Implication***:
Please note that $\to$ on the right-hand side of the definition below is the implication operation on $\mathcal{L}$.
$$(Q \to R)(x, y) = Q(x, y) \to R(x, y)$$
6. ***Inclusion***:
The inclusion $\subseteq$, which is induced by the intersection or union of $\mathcal{L}$-fuzzy relations, has to be read as follows [61]:
$$\mathrm{Q} \sqsubseteq \mathrm{R} \Leftrightarrow \forall\ \mathrm{x} \in \mathrm{A}\ ,\mathrm{y} \in \mathrm{B} : \mathrm{Q(x,y)} \leq \mathrm{R(x,y)}.$$
Again, this is a generalization of $\subseteq$ defined for regular relations.

[8] Conversion is known as Transposition as well.

7. ***Empty and Universal relations***:
   The set of all $\mathcal{L}$-fuzzy relations between A and B with least and greatest element are defined by:

$$\perp\!\!\!\perp(x, y) = 0, \quad \top(x, y) = 1.$$

8. ***Identity relation***:
   The identity relation on the set A can be shown by:

$$\mathbb{I}(x, y) := \begin{cases} 1 & \textit{iff} \quad x = y, \\ 0 & \textit{otherwise}. \end{cases}$$

Some other main combined operations are defined as follows [61]:

Definition 2.3.2.

Left Residual:
Let consider some other principal operations on $\mathcal{L}$-Fuzzy relations Q: X $\rightarrow$ Y and M: X $\rightarrow$ K, then Q \ M: Y $\rightarrow$ K:

$$(Q \setminus M)(y, k) = \bigwedge_{x \in X} Q(x, y) \longrightarrow M(x, k)$$

where the $\longrightarrow$ represents the relative *pseudocomplement*.

Right Residual:
Let consider $\mathcal{L}$-Fuzzy relations Q: X $\rightarrow$ Y and T: V $\rightarrow$ Y, then Q / T: X $\rightarrow$ V:

$$(Q \,/\, T)(x, v) = \bigwedge_{y \in Y} T(v, y) \longrightarrow Q(x, y)$$

where the $\longrightarrow$ represents the relative *pseudocomplement*.

The math representation of residuals in fuzzy relations are the same as residuals in boolean relations. The only difference is that *universal quantifier* is replaced by *meet* operation in both residuals in fuzzy area.

Symmetric Quotient:

Let consider $\mathcal{L}$-Fuzzy relations T: V $\rightarrow$ Y and S: V $\rightarrow$ N, then syq(T, S):

$$syq(T,S)(y,n) = \bigwedge_{v \in V} T(v,y) \longleftrightarrow S(v,n)$$

Comparing ϵ relation in the area of $\mathcal{L}$-fuzzy relations with boolean relations, this time $\mathcal{P}$(A) is the set of all *fuzzy* subsets. $\mathcal{L}$-fuzzy subset M of a set A is simply a function M: A $\rightarrow \mathcal{L}$. Now, we have the definition of ϵ as follows:

Definition 2.3.3. The definition of ϵ: A $\rightarrow \mathcal{P}$(A) in the area of $\mathcal{L}$-fuzzy relations can be defined by [1]:

$$\epsilon(\mathrm{x},\mathrm{M}) = \mathrm{M}(\mathrm{x}).$$

In the area of $\mathcal{L}$-fuzzy relations, we also have some properties of ϵ relation. Since two new operations called ***support*** or ***up-arrow*** ($\uparrow$) and ***kernel or down-arrow*** ($\downarrow$) need to be introduced in such properties, the mentioned properties are represented in Lemma. 2.3.2.

Also, we have a few more properties shown in the following lemma.

Lemma 2.3.1. Let consider $\mathcal{L}$-fuzzy relations Q,Q′: A $\rightarrow$ B, R: B $\rightarrow$ C, S: C $\rightarrow$ D and T: A $\rightarrow$ C, then we have [61]

1. Q ; $\mathbb{I}_B$ = Q and $\mathbb{I}_B$; R = R,
2. $(Q;R);S = Q;(R;S)$,
3. $(Q \sqcap Q')^{\smile} = Q^{\smile} \sqcap Q'^{\smile}$,
4. $(Q;R)^{\smile} = R^{\smile};Q^{\smile}$,
5. $(Q^{\smile})^{\smile} = Q$,
6. Q ; R $\sqcap$ T $\sqsubseteq$ Q ; (R $\sqcap$ $Q^{\smile}$; T),
7. $Q;\perp\!\!\!\perp_{BC} = \perp\!\!\!\perp_{AC}$.

Proofs of all assertions can be found in [61].

Considering the following $\mathcal{L}$-Fuzzy relations and lattice $\mathcal{L}$, we clarify the most complex operations defined by using some examples. Examples of the union and intersection operations are skipped. union($\cup$) and intersection($\cap$) operations are performed using meet ($\wedge$) and join($\vee$) respectively on $\mathcal{L}$-Fuzzy relations and the join and meet between two elements are simply their maximum and minimum elements.
Identically, on the basis of the meet operation, the implication between two relations can be fulfilled in a straightforward way.
Also, it is worthwhile to mention that the composition of $\mathcal{L}$-fuzzy relations is similar as matrix multiplication in linear algebra in terms of implementation. Meet and join are used instead of addition and multiplication, respectively.

Example 2.3.3. Let Q, M, T, and S be relations with the same dimensions and $\mathcal{L}$ = [0,1]. As already mentioned, *join* and *meet* are *greatest* (*max*) and *smallest* (*min*) elements, respectively. Also, *implication* ($\rightarrow$) on $\mathcal{L}$-fuzzy relations is x$\rightarrow$y = y iff y < x and x$\rightarrow$y = 1 iff x $\leq$ y.

$$\boldsymbol{Q} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} y_1 & y_2 & y_3 & y_4 \end{array} \\ \begin{pmatrix} 0.75 & 0.25 & 1 & 0.75 \\ 1 & 0.50 & 0.75 & 0.25 \\ 0 & 1 & 0.25 & 0 \\ 0.50 & 0.25 & 0.50 & 1 \end{pmatrix} \end{array}, \quad \boldsymbol{M} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} k_1 & k_2 & k_3 & k_4 \end{array} \\ \begin{pmatrix} 0 & 0.75 & 0.50 & 0.25 \\ 1 & 1 & 0.25 & 0.75 \\ 0.25 & 0.50 & 0.25 & 1 \\ 0.25 & 0 & 0.25 & 0 \end{pmatrix} \end{array}$$

$$\boldsymbol{T} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} y_1 & y_2 & y_3 & y_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0.25 & 1 \\ 1 & 0.75 & 0.75 & 0.25 \\ 1 & 1 & 1 & 0.50 \\ 0.50 & 1 & 0.50 & 1 \end{pmatrix} \end{array}, \quad \boldsymbol{S} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} n_1 & n_2 & n_3 & n_4 \end{array} \\ \begin{pmatrix} 0.25 & 0 & 0.25 & 1 \\ 0.75 & 0.75 & 1 & 0.25 \\ 1 & 1 & 1 & 0.50 \\ 0.50 & 1 & 0.75 & 1 \end{pmatrix} \end{array}$$

- *Implication*

$$Q \rightarrow M = \begin{pmatrix} 0 & 1 & 0.50 & 0.25 \\ 1 & 1 & 0.25 & 1 \\ 1 & 0.50 & 1 & 1 \\ 0.25 & 0 & 0.25 & 0 \end{pmatrix}$$

- *Composition*

$$Q;M \;=\; \begin{pmatrix} 0.25 & 0.75 & 0.50 & 1 \\ 0.50 & 0.75 & 0.50 & 0.75 \\ 1 & 1 & 0.25 & 0.75 \\ 0.25 & 0.50 & 0.50 & 0.50 \end{pmatrix}$$

As I mentioned before, the composition of $\mathcal{L}$-fuzzy relations is similar as *matrix multiplication* in *linear algebra* in terms of implementation. The only difference is that *Meet* and *join* are used instead of *multiplication* and *addition*, respectively.

As an example to calculate the value of 0.75 located in the row number 0 and column number 1 of the composition matrix, we have to consider row number 0 of matrix Q and column number 1 of matrix M. Then we have:

$$(0.75 \bigwedge 0.75) \bigvee (0.25 \bigwedge 1) \bigvee (1 \bigwedge 0.5) \bigvee (0.75 \bigwedge 0) = 0.75$$

- *Left Residual*

$$Q \setminus M \;=\; \begin{array}{c} \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{array} \begin{array}{c} \begin{array}{cccc} k_1 & k_2 & k_3 & k_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0.25 & 0 \end{pmatrix} \end{array}$$

In calculation of the left residual between two fuzzy relations, the matrix representation is the same as calculation of left residual between boolean relations. The only difference is that the entry in the left residual matrix is the largest lattice value $\mathcal{L}$ so that the calculation of meet among the values in the column of the first matrix and $\mathcal{L}$ is smaller and equal to corresponding value in the column of the second matrix. As an example, the value of element located in row 0 and column 0 in $Q \setminus M$ is calculated as follows:

$0.75 \wedge 0 = 0 \leq 0$, $1 \wedge 1 = 1 \leq 1$, $0 \wedge 1 = 0 \leq 0.25$, and $0.5 \wedge 0.25 = 0.25 \leq 0.25$.

Then we have: $0 \wedge 1 \wedge 1 \wedge 0.25 = 0$. So, 0 is the value of the element in row 0 and column 0 of the left residual matrix.

- *Right Residual*

$$Q/T \;=\; \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 0.25 & 0.50 & 0.50 \\ 0 & 1 & 1 & 0.50 \\ 0 & 0.75 & 1 & 1 \\ 0 & 0.25 & 0.50 & 1 \end{pmatrix} \end{array}$$

  In calculation of the right residual between two fuzzy relations, the matrix representation is the same as calculation of left residual between fuzzy relations. The only difference is that we have to consider rows instead of columns to compute it.

- *Symmetric Quotient*

$$syq(T, S) \;=\; \begin{array}{c} \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{array} \begin{array}{c} \begin{array}{cccc} n_1 & n_2 & n_3 & n_4 \end{array} \\ \begin{pmatrix} 0.25 & 0 & 1 & 0.25 \\ 0 & 1 & 0 & 0 \\ 0.25 & 0 & 0.50 & 0.25 \\ 0.25 & 0 & 0.25 & 0.25 \end{pmatrix} \end{array}$$

2.3.2 T-norm Like (*-based) Operations on $\mathcal{L}$-Fuzzy Relations

We can define *-based operations as:

$$\text{star: } * \quad \text{starImpl: } \twoheadrightarrow \quad \text{starResidual: } \not* \quad \text{starComposition: } ;^*$$

Generally, a t-norm like operation on a *Heyting algebra* $\mathcal{L}$ is a straight-forward *generalization* of t-norms on the unit interval [0, 1].

Definition 2.3.4. Let $\mathcal{L}$ be a *Heyting algebra* together with a ***t-norm like operation*** $*$ as the lattice of truth values instead of the unit interval [0, 1], i.e., structure $\langle\, \mathcal{L}, * \,\rangle$ with an operation $*$ on $\mathcal{L}$, then we can define the following $*$-based operations on relations [1]:

1. $(Q * R)(a, b) = Q(a, b) * R(a, b)$,
2. $(Q \twoheadrightarrow R)(a, b) = Q(a, b) \twoheadrightarrow R(a, b)$,
3. $(Q \;;^* S)(a, c) \;=\; \bigvee_{b \in B} Q(a, b) * S(b, c)$,
4. $(T \not* S)(a, c) \;=\; \bigwedge_{c \in C} S(b, c) \twoheadrightarrow T(a, c)$.

2.3.3 Crispness and Scalar

Among the $\mathcal{L}$-fuzzy relations two subclasses are of interest; Crisp relations and Scalar relations. A relation R is called 0–1 *crisp* iff R(x, y) = 0 or R(x, y) = 1 holds for all x and y. Crisp relations may be identified with classical (binary) relations [62]. Therefore, The crisp relation can be identified with regular relations that is the regular true and false of $\mathbb{B}$. Transformation of a fuzzy relation to crisp relation is required to define two operations called *support* (↑) and *kernel* (↓). These operations are also known as the *up-arrow* and *down-arrow* respectively. The up-arrow operation increases the membership degrees which are greater than 0, whereas the down-arrow operation decreases the membership degrees which are smaller than 1.

Definition 2.3.5. Suppose Q: X × Y is a fuzzy relation. Then $Q^{\uparrow}$ and $Q^{\downarrow}$ can be defined as:

$$Q^{\uparrow}(x,y) := \begin{cases} 1 & \textit{iff} \quad Q(x,y) \neq 0, \\ 0 & \textit{otherwise}. \end{cases}$$

$$Q^{\downarrow}(x,y) := \begin{cases} 1 & \textit{iff} \quad Q(x,y) = 1, \\ 0 & \textit{otherwise}. \end{cases}$$

Example 2.3.4. Let Q be a fuzzy relation. After applying two fuzzy-crisp transformations on it, we have:

$$\boldsymbol{Q} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccccc} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \end{array} \\ \begin{pmatrix} 0.75 & 0 & 0.25 & 1 & 0.50 & 1 \\ 0.50 & 0.25 & 0.25 & 1 & 0.20 & 0 \\ 1 & 0 & 0.75 & 0 & 1 & 1 \\ 0.25 & 0.25 & 0 & 0.75 & 1 & 0.50 \\ 1 & 0.75 & 0.50 & 1 & 0.75 & 1 \\ 0.25 & 0 & 0.25 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Using support and kernel operations applied on the Q relation, the following crisp relations are returned respectively:

$$Q^{\uparrow} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccccc} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$Q^{\downarrow} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccccc} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Lemma 2.3.2. The properties of the boolean relations for ϵ relation change slightly to have the properties in fuzzy relations [1] as follows:

- $syq(\epsilon, \epsilon)^{\downarrow} = \mathbb{I}$
- $syq(R, \epsilon)^{\downarrow}$ is *total* for every R.

For a comparison of the definitions among properties of ϵ relations for boolean and fuzzy relations, and also the reason for using the two above assertions presented here in the case of arrow category including down arrows, we refer to [64].

In the next theorem we have summarized some properties of the operation defined above [61].

Theorem 2.3.1. Let $\mathcal{L}$ be a complete Brouwerian lattice (Heyting algebra), and Q,R: A $\rightarrow$ B and S: B $\rightarrow$ C be $\mathcal{L}$-fuzzy relations. Then we have

(i) Q is *crisp* iff $Q^{\uparrow}$ = Q iff $Q^{\downarrow}$ = Q,

(ii) $(R^{\smile}; S^{\downarrow})^{\uparrow} = (R^{\uparrow})^{\smile}; S^{\downarrow}$,

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

All proofs not given in this section are straightforward and can be found in [61].

2.3.3.1 Scalar Relations

There are several possibilities to identify a class of $\mathcal{L}$-fuzzy relations with the lattice $\mathcal{L}$ itself, e.g., one could choose ideal elements. In our approach we will take *scalar* relations. The lattice $\mathcal{L}$ may also be characterized by sub-identities that use exactly one element from the lattice $\mathcal{L}$ to relate each element to itself. Such a relation is called a *scalar*.

An $\mathcal{L}$-fuzzy relation α_A^u: A $\rightarrow$ A is called a *scalar* on A induced by $u \in L$ if [61, 62]:

$$\alpha_A^u(x, y) := \begin{cases} u & \textit{iff} \quad x = y, \\ 0 & \quad\quad x \neq y. \end{cases}$$

In other words, a relation α: A $\rightarrow$ A is called a scalar on A [30, 42]:

$$\pi_{AA}; \alpha = \alpha ; \pi_{AA} \qquad \textit{iff} \qquad \alpha \sqsubseteq \mathbb{I}_A.$$

Therefore, in the matrix form, the scalar (fuzzy) defined above has the form of:

$$\begin{pmatrix} u & 0 & 0 & \cdots \\ 0 & u & 0 & \cdots \\ 0 & 0 & u & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

2.4 Types of Categories

The calculus of binary relations plays an important role in the development of logic and algebra. Furthermore, in computer science categories of binary relations are used to model programs as well as specifications and properties of programs in the same language. There have been several attempts [43] to extend the categorical approach from binary to $\mathcal{L}$-fuzzy relations [62]. In this section, Our purpose is to introduce multiple types of categories and allegories that are needed to derive our algebraic framework for $\mathcal{L}$-fuzzy relations.

2.4.1 Categories

To be able to make hidden properties of an object exposed to use them for the applications of new methods, we can use the benefits of categories. Usually a binary relation acts

between two different sets. Therefore, an algebraic theory for relations should reflect this kind of typing, i.e., the theory should have a suitable notion of source and target of its elements. A convenient framework for that is given by category theory [61]. We write R: A $\rightarrow$ B to indicate that a morphism R of a category R has source A and target B. The collection of all morphisms R: A $\rightarrow$ B is denoted by R[A,B]. And, the identity morphism on A is written as $\mathbb{I}_A$ [66].
Another way of definition for category theory is that it proposes a set of tools to give legal or formal status to the mathematical concepts and formations. In this approach, it can take advantage of a directed graph with labels to explain the structure and theories in mathematics which is called category. The nodes and directed labeled edges of the graph are called objects and morphisms respectively.
In this chapter, some basic notions of category theory are explained. [6, 61, 62] comprise comprehensive introductions to this theory. Also, a more formal definition of the category of $\mathcal{L}$-fuzzy relation can be found in [62] specifically.

Definition 2.4.1. A category C consists of [61]:

1. a class of objects Obj_C,
2. a class of morphisms C[A,B] for every pair of objects A and B,
3. an associative binary (partial) operation; mapping each pair of morphisms f in C[A,B] and g in C[B,C] to a morphism f;g in C[A,C],
4. for every object A a morphism $\mathbb{I}_A$ such that for all f in C[A,B] and g in C[C,A] we have $\mathbb{I}_A$; f = f and g;$\mathbb{I}_A$ = g.

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

The following table lists some common categories by specifying their objects and morphisms [61]:
The standard model of these categories is the category **Rel** of sets and relations, of course. However, there are many other models. For example, given a complete Heyting algebra one may form the category **Rel**($\mathcal{L}$) of sets and $\mathcal{L}$-valued (or $\mathcal{L}$-fuzzy) relations. Such a relation R: A $\rightarrow$ B is a function from A $\times$ B to $\mathcal{L}$ assigning to each pair in A $\times$ B a degree of membership in the relation R from Heyting algebra $\mathcal{L}$ [63].

To clarify **Rel** and $\mathcal{L}$-**Rel**, notice that the category **Rel** is a special case of the category

[9] Or it may be indicated by **Rel**($\mathcal{L}$).

Table 2.2: Lists of some common categories by specifying their objects and morphisms.

Category	Objects	Morphisms
Set	sets	functions
Rel	sets	binary relations
[0, 1]-**Rel**	nonempty sets	fuzzy relations
$\mathcal{L}$-**Rel** [9]	nonempty sets	$\mathcal{L}$-fuzzy relations
PO	posets	monotone functions
$\mathbf{Vct}_{\mathbb{F}}$	vector spaces over the field F	linear functions

$\mathcal{L}$-**Rel** of $\mathcal{L}$ relations if $\mathcal{L}$ is the Boolean algebra $\mathbb{B}$ of the truth values, i.e., we can use any of the notations **Rel**, **Rel**($\mathbb{B}$), or **Rel**($\{0, 1\}$) for the category of sets and binary relations [66]. Obviously, the collection of crisp relations in **Rel**($\mathcal{L}$) or $\mathcal{L}$-**Rel** is equivalent to **Rel** [63]. Also, [0, 1]-**Rel** is the notation to show the the category of sets and fuzzy relations. As mentioned earlier, a morphism in C[X, Z] can be converted as a relational form of r: X $\rightarrow$ Z. A diagram can illustrate this transformation where nodes and directed labeled arrows represent objects and morphisms in the category respectively. As an apprehensible example, consider morphisms of p: X $\rightarrow$ Y, q: Y $\rightarrow$ Z and r: X $\times$ Z in category C. Fig. 2.5 illustrates the composition between p and q which is equal to h. i.e. p; q = r.

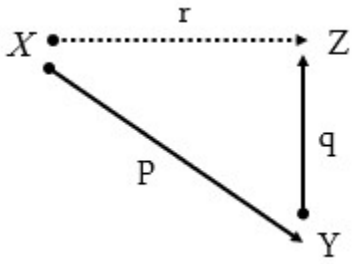


Figure 2.5: Visual representation of categories.

2.4.2 Allegories

Allegories provide a suitable abstract framework to reason about binary relations [28, 53, 66]. In simple words, allegories are referred to categories fulfilling a certain structural category **Rel** of sets and a binary relation between them as morphisms. It is worth noting that all the basic definitions and properties of allegory are taken from [61] in this section. The definition of allegory can be generalized as follows:

Definition 2.4.2. An allegory ***R*** is a category satisfying the following:

1. For all objects A and B the class ***R***[**A**,**B**] is a lower semilattice. 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 $^\smile$ (called the converse operation) such that for all relations Q, R: A $\to$ B and S: B $\to$ C the following holds:

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

3. For all relations Q: A $\to$ B and R,S: B $\to$ C we have:

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

4. For all relations Q: A $\to$ B, R: B $\to$ C, and S: A $\to$ C, the modular law holds as follows:

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

The category $\mathcal{L}$-Rel of $\mathcal{L}$-fuzzy relations with meet $\cap$ and conversion $^\smile$ is an allegory.

Lemma 2.4.1. Let *R* be an allegory, A,B,C 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) $(\mathbb{I}_A)^\smile = \mathbb{I}_A$,
(2) $(Q \sqcap R); S \sqsubseteq Q; S \sqcap R; S$,
(3) ; is monotone in both arguments,
(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) $\mathbb{I}_A \sqcap (U \sqcap V); (U \sqcap V)^\smile = \mathbb{I}_A \sqcap U; V^\smile = \mathbb{I}_A \sqcap V; U^\smile$,
(8) $Q = (\mathbb{I}_A \sqcap Q; Q^\smile); Q = Q; (\mathbb{I}_B \sqcap Q^\smile; Q)$.

Definition 2.4.3. Let *R* be an allegory and Q: A $\to$ B. Then we call:

(1) Q a *univalent* iff $Q^\smile$; Q $\sqsubseteq \mathbb{I}_B$,
(2) Q a *total* iff $\mathbb{I}_A \sqsubseteq Q; Q^\smile$
(3) Q a *map* iff Q is univalent and total,

(4) Q a *injective* iff $Q^\smile$ is univalent,
(5) Q a *surjective* iff $Q^\smile$ is total,
(6) Q a *bijective* iff $Q^\smile$ is a map,
(7) Q a *bijection* iff Q is a bijective map.

The class of univalent relations, the class of total relations, and, hence, the class of mappings is closed under composition. For the univalent relations, we have:

$$(Q;R)^\smile;Q;R = R^\smile;Q^\smile;Q;R$$

Some other properties of univalent relations are summarized in the next lemma.

Lemma 2.4.2. Let *R* be an allegory, Q: A → B be univalent and R, S: B → C, T: C → A, and U: C → B. Then we have:

(1) Q ;(R ⊓ S) = (Q ; R) ⊓ (Q ; S),
(2) T ; Q ⊓ U = (T ⊓ U ; Q) ; Q.

All properties and proofs, including those not introduced in this section regarding categories, allegories, and related structures, can be found comprehensively in [28, 61, 62, 63].

2.4.2.1 Distributive Allegories

Consider the collection of binary relations on a fixed set. This structure constitutes a distributive lattice with a least element [61].
With reference to Table. 2.2, category of $\mathcal{L}$-fuzzy relations which is known as ($\mathcal{L}$-**Rel**) combining with meet and converse operations along with the collection of binary relations on fixed sets generates a distributive lattice including a smallest element. This is our incentive to replace lower semilattices by distributive lattices as the basic order structure.

Definition 2.4.4. A distributive allegory *R* is an allegory satisfying the following [61]:

1. The classes *R*[A,B] are distributive lattices with a least element. Union and the least element are denoted by ⊔, $\bot_{AB}$, respectively.
2. For all relations Q: A → B we have Q ; $\bot_{BC} = \bot_{AC}$.
3. For all relations Q: A → B and R, S: B → C, we have Q ; (R ⊔ S) = (Q ; R) ⊔ (Q ; S).

Based on the above definition, the allegory $\mathcal{L}$-Rel of $\mathcal{L}$-fuzzy relations with the basic set operation union is distributive.

Lemma 2.4.3. Let *R* be a distributive allegory. Then for all Q, R: A → B and S: B → C we have [61]:

1. $(\bot_{AB})^\smile = \bot_{BA}$,
2. $\bot_{CA}; Q = \bot_{CB}$,
3. $(Q \sqcup R)^\smile = Q^\smile \sqcup R^\smile$,
4. $(Q \sqcup R)$; S = (Q ; S) ⊔ (R ; S).

As above, all proofs are discussed in [61] clearly.

2.4.2.2 Division Allegories

The next step in the hierarchy of allegories are Division allegories. A Division allegory can be generated by addition of the residual (also known as division or adjoint) operation to a Distributive allegory. Division allegories are characterized by the fact that *left residual* and *right residual* are the same as *upper left adjoint* and *upper right adjoint*, respectively. These definitions are detailed more in [28, 61]:

Definition 2.4.5.
A Division allegory *R* is a Distributive allegory such that ; has an *upper left adjoint*, i.e., for all relations R: B → C and S: A → C there is a relation S/R: A → B (called the ***left residual*** of S and R) such that for all Q: A → B the following holds:

$$\text{Q ; R} \sqsubseteq \text{S} \Leftrightarrow \text{Q} \sqsubseteq \text{S/R}. \qquad (*)$$

Considering the above relations, in a Division allegory there is also an *upper right adjoint* for ;, which will be denoted by Q\S and called the ***right residual*** of S and Q as below:

$$Q \setminus S = (S^\smile / Q^\smile)^\smile$$

Consider the computation:

$$\text{Q ; R} \sqsubseteq \text{S} \Leftrightarrow R^\smile ; Q^\smile \sqsubseteq S^\smile$$

$$\Leftrightarrow R^\smile \sqsubseteq S^\smile / Q^\smile \qquad (*)$$

$$\Leftrightarrow \text{R} \sqsubseteq (S^\smile / Q^\smile)^\smile$$

The above computation shows that the operation $Q \setminus S$ satisfies a similar condition than the other residual.

Another operation can be formed using a combination of both left and right residuals in the Division allegories. The new operation is called the *symmetric version*. The ***symmetric quotients of the residuals*** may be defined as [61]

$$\text{syq (Q, R)} := (\text{Q} \setminus \text{R}) \sqcap (Q^{\smile}/R^{\smile}).$$

By definition, this relation is the greatest solution X of the inclusions

$$\text{Q ; X} \sqsubseteq \text{R} \qquad \text{and} \qquad \text{X ; } R^{\smile} \sqsubseteq Q^{\smile}.$$

In the next lemma we have denoted some *properties of the residuals* [61].

Lemma 2.4.4. Let R be a division allegory and Q, Q1, Q2: A $\rightarrow$ B, R, R1, R2: B $\rightarrow$ C, and S, S1, S2: A $\rightarrow$ C. Then we have :

1. $Q \sqsubseteq (Q;R)/R$ and $R \sqsubseteq Q \setminus (Q;R)$,
2. $(S/R);R \sqsubseteq S$ and $Q;(Q \setminus S) \sqsubseteq S$,
3. $S/(Q \setminus S) \sqsubseteq Q$ and $(S/R) \setminus S \sqsubseteq R$,
4. $Q_2 \sqsubseteq Q_1, R_2 \sqsubseteq R_1$ and $S_1 \sqsubseteq S_2$ implies $S_1/R_1 \sqsubseteq S_2/R_2$ and $Q_1 \setminus S_1 \sqsubseteq Q_2 \setminus S_2$,
5. $(S_1 \sqcap S_2)/R = (S_1/R) \sqcap (S_2/R)$ and $Q \setminus (S_1 \sqcap S_2) = (Q \setminus S_1) \sqcap (Q \setminus S_2)$,
6. $S/(R_1 \sqcup R_2) = (S/R_1) \sqcap (S/R_2)$ and $(Q_1 \sqcup Q_2) \setminus S = (Q_1 \setminus S) \sqcap (Q_2 \setminus S)$.

In the next lemma, we have summarized some basic *properties of symmetric quotients* found in both [61, 10].

Lemma 2.4.5. Let R be a division allegory, Q: A $\rightarrow$ B, R: A $\rightarrow$ C, S: A $\rightarrow$ D be relations, and f: D $\rightarrow$ A be 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).

As before, all proofs are shown in [61].

Now, we can describe the basic structure in our approach. It consists of three categories: *Heyting*, *arrow*, and *fuzzy* categories. Detailed explanations are provided in the following sections for all three categories.

2.4.3 Dedekind Category and Properties (Heyting Category)

There have been several attempts (see [43]) to extend the categorical approach from binary to $\mathcal{L}$-fuzzy relations. It turned out that all approaches lack a sufficient definition of crispness, which is fundamental in those theories. In [60] it was shown that the language of Dedekind categories is too weak to formalize that notion [62]. We will focus on a solution for this weakness later in the next section.
Among all categorical approaches, we will focus on a crucial one known as Dedekind category in this section. Allegories, and Dedekind categories in particular, are a fundamental tool to reason about relations [49] and are widely used categorical versions of the calculus of binary relations [62]. Categories of this type are called locally *complete division allegories* in [28, 65]. It means that a Dedekind category is typically a division allegory which is a complete distributive allegory or where the distributive lattices are complete Heyting algebras [28]. We will use the framework of Dedekind categories as a basic theory of relations [49].

Definition 2.4.6. A Dedekind category D is a division allegory so that every D[A,B] is a *complete Brouwerian lattice*. The greatest element in D[A,B] is denoted by $\top_{AB}$ [61].

Now, some properties satisfying by a Dedekind category D are summarized as follows [65]

Definition 2.4.7.

1. For all objects A and B the collection D[A, B] is a *complete distributive lattice*. *Meet*, *join*, the *induced ordering*, the *least*, and the *greatest element* are denoted by $\sqcap$, $\sqcup$, $\sqsubseteq$, $\bot_{AB}$, $\top_{AB}$, respectively.

2. There is a monotone operation $^\smile$ (called *converse*) mapping a relation Q: $A \to B$ to $Q^\smile$: $B \to A$ such that for all relations Q: $A \to B$ and R: $B \to C$ the following holds:

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

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

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

4. For all relations R: $B \to C$ and S: $A \to C$, there is a relation S/R : $A \to B$ (called the *left residual* of S and R) such that for all X: $A \to B$ the following holds:

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

5. Based on the residual operation in the definition above it is possible to define a *right residual* by $R \setminus S = (S^\smile/R^\smile)^\smile$. This operation is characterized by:

$$\text{R ; X} \sqsubseteq \text{S} \Leftrightarrow \text{X} \sqsubseteq \text{R} \setminus \text{S}.$$

For the sake of making the matter more clear, some further properties of Dedekind categories are indicated as:

Definition 2.4.8. Let D be a Dedekind category, Q: A $\to$ B, R: A $\to$ D, S: B $\to$ C be relations and U: A $\to$ A be a partial identity, i.e., U $\sqsubseteq \mathbb{I}_A$. Then we have [65]:

1. $(\text{Q} \sqcap \text{R} \ ; \pi_{DB}) \ ; \text{S} = \text{Q} \ ; \text{S} \sqcap \text{R} \ ; \pi_{DC}$,
2. $\text{S} \ ; (\pi \ ; \text{R} \sqcap \text{Q}) = \pi \ ; \text{R} \sqcap \text{S} \ ; \text{Q}$,
3. $\text{U} \ ; \text{Q} = \text{U} \ ; \pi_{AB} \sqcap \text{Q}$.

All proofs can be found in [53, 54, 61].

It is noteworthy that for every valid universally quantified equation or inclusion its dual (or opposite) by reversing the direction of composition is also valid. For example, the dual of the Definition. 2.4.8 (1) which is shown in (2) is also valid for appropriate relations Q, R, and S.
For further basic properties of relations in a Dedekind category we refer the reader to [23, 28, 60, 62].
In the following lemma, some basic properties of relations in Dedekind category are collected [60]. We will use such properties throughout the paper and in our implementation.

Lemma 2.4.6. Let D be a Dedekind category, let A, B, C be objects of D, and for i ∈ I, let Q_1, Q_2: A → B and R_1, R_2: B → C be relations. Then:

1. $\amalg_{AB}^{\smile} = \amalg_{BA}$, $\pi_{AB}^{\smile} = \pi_{BA}$, and $\mathbb{I}_A^{\smile} = \mathbb{I}_A$,
2. $\pi_{AA} \,;\, \pi_{AB} = \pi_{AB} \,;\, \pi_{BB} = \pi_{AB} \,;\, \pi_{BA} \,;\, \pi_{AB} = \pi_{AB}$,
3. $Q_1 \,;\, \amalg_{BC} = \amalg_{AB} \,;\, R_1 = \amalg_{AC}$,
4. if $Q_1 \sqsubseteq Q_2$ and $R_1 \sqsubseteq R_2$ then $Q_1 \,;\, R_1 \sqsubseteq Q_2 \,;\, R_2$,
5. $Q_1 \sqsubseteq Q_1 \,;\, Q_1^{\smile} \,;\, Q_1$.

All proofs for Lemma. 2.4.6 are described in detail in [61]. Since $\mathcal{L}$–Rel[A, B] is a *complete Heyting algebra*, we can obviously consider that $\mathcal{L}$-Rel is a Dedekind category. Whereas, as mentioned before, crispness is not covered to define as a relation property in Dedekind category. Therefore, the necessity of introducing another category is felt. This new form of category is known as *Arrow category*. To achieve the crispness property of $\mathcal{L}$-fuzzy relations, up ($^{\uparrow}$) and down($^{\downarrow}$) arrows are defined as new structures in this category [62].

2.4.4 Arrow category and Properties

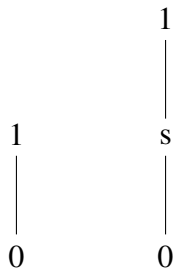
As already mentioned, we need an additional concept to define a suitable algebraic theory of $\mathcal{L}$-fuzzy relations due to the lack of crispness. Now, our focus is on mapping every relation to its support ($^{\uparrow}$) and kernel ($^{\downarrow}$), respectively. In other terms, those operations map the relation to the greatest 0–1 crisp relation it contains and to the least 0–1 crisp relation it is included in [61]. We now give an abstract definition [61, 62, 66].

Definition 2.4.9. An *arrow category* A is a Dedekind category with $\amalg_{AB} \neq \pi_{AB}$ for all objects A and B together with two operations $^{\uparrow}$ and $^{\downarrow}$ satisfying the following:

1. $R^{\uparrow}, R^{\downarrow}$: A → B for all R: A → B.
2. $Q^{\uparrow} \sqsubseteq$ R *iff* Q $\sqsubseteq R^{\downarrow}$. So, ($^{\uparrow}$, $^{\downarrow}$) is a *Galois* correspondence.
3. $(R^{\smile} \,;\, S^{\downarrow})^{\uparrow} = R^{\uparrow\smile} \,;\, S^{\downarrow}$ for all R: B → A and S: B → C.
4. $(Q \sqcap R^{\downarrow})^{\uparrow} = Q^{\uparrow} \sqcap R^{\downarrow}$ for all Q, R: A → B.
5. if $\alpha \neq \amalg_{AA}$ is a non-zero scalar then $\alpha^{\uparrow} = \mathbb{I}_A$.

2.4.4.1 Crispness in Arrow Category

The two new operations $()^{\uparrow}$ and $()^{\downarrow}$ compute the support and the kernel of a relation, i.e., $R^{\uparrow}$ is the smallest crisp relation that contains R, and $R^{\downarrow}$ is the largest crisp relation that is contained in R. We want to explain this fact by using the matrix model below. Axiom (5) clearly indicates that $()^{\uparrow}$ maps all non-zero elements from $\mathcal{L}$ to 1 [66].
Relations between finite sets can be represented by matrices [59]. As an example [66], let us consider two sets of criteria for cars. A person might *like* (1) or *dislike* (0) standard transmission in a specific car. In addition, a person may rate the maximum speed of a car as *insufficient* (0), *sufficient* (s), or *superior* (1). In other words, we have two *Heyting algebras* given by the Hasse diagrams shown as below:



In the following matrix, we take their Cartesian product (which is again a *Heyting algebra*) as *L*.

Now, given a set of *persons* P = {Dave, Elizabeth, John} and of *cars* C = {BMW, Dodge, Fiat, Ferrari, Mercedes, Porsche} we may define the relation $\mathcal{L}$: P → C of a person liking a specific car by the following matrix:

$$\boldsymbol{L} = \begin{array}{c} \\ \textit{Dave} \\ \textit{Elizabeth} \\ \textit{John} \end{array} \begin{array}{c} \begin{array}{cccccc} \textit{BMW} & \textit{Dodge} & \textit{Fiat} & \textit{Ferrari} & \textit{Mercedes} & \textit{Porsche} \end{array} \\ \begin{pmatrix} (0,s) & (1,0) & (0,0) & (0,1) & (1,s) & (1,1) \\ (0,1) & (1,1) & (0,s) & (0,1) & (1,1) & (0,1) \\ (1,s) & (0,0) & (0,0) & (1,1) & (1,1) & (1,1) \end{pmatrix} \end{array}$$

Each row and each column of the matrix above corresponds to a person and a car, respectively. For example, the value (0,s) in the second row and third column indicates that

Elizabeth does not like the transmission of the Fiat but rates its speed as sufficient.

With respect to the explanation of the previous paragraph we obtain:

Example 2.4.1.

$$L^{\downarrow} = \begin{array}{c} \\ Dave \\ Elizabeth \\ John \end{array} \begin{array}{c} \begin{array}{cccccc} BMW & Dodge & Fiat & Ferrari & Mercedes & Porsche \end{array} \\ \begin{pmatrix} (0,0) & (0,0) & (0,0) & (0,0) & (0,0) & (1,1) \\ (0,0) & (1,1) & (0,0) & (0,0) & (1,1) & (0,0) \\ (0,0) & (0,0) & (0,0) & (1,1) & (1,1) & (1,1) \end{pmatrix} \end{array}$$

$$L^{\uparrow} = \begin{array}{c} \\ Dave \\ Elizabeth \\ John \end{array} \begin{array}{c} \begin{array}{cccccc} BMW & Dodge & Fiat & Ferrari & Mercedes & Porsche \end{array} \\ \begin{pmatrix} (1,1) & (1,1) & (0,0) & (1,1) & (1,1) & (1,1) \\ (1,1) & (1,1) & (1,1) & (1,1) & (1,1) & (1,1) \\ (1,1) & (0,0) & (0,0) & (1,1) & (1,1) & (1,1) \end{pmatrix} \end{array}$$

Definition 2.4.10. $\mathcal{L}$-Rel with $^{\uparrow}$ and $^{\downarrow}$ is an *arrow* category [61].

Definition 2.4.11. A relation R: A $\rightarrow$ B of an arrow category is called ***crisp*** iff $R^{\uparrow}$ = R (or equivalently $R^{\downarrow}$ = R) [62].

We now turn to another definition of Arrow category. The following definition and its proof are presented in [61]:

Definition 2.4.12. Let $\mathcal{L}$ be a complete Brouwerian lattice with $0 \neq 1$. Then $\mathcal{L}$-Rel together with $^{\uparrow}$ and $^{\downarrow}$ is an *arrow category*.

2.4.4.2 Properties in Arrow Category

Basic Properties of Relations in Arrow Category In the next lemma, we can found a comprehensive collection of basic properties of relations in arrow categories used in this paper [61]. The complete proof for the Lemma. 2.4.7 can also be found in the same paper.

Lemma 2.4.7. Let $\mathcal{A}$ be an arrow category and Q,R: A $\rightarrow$ B, S: B $\rightarrow$ C, T: A $\rightarrow$ C. Then we have:

1. $\mathbb{I}_A^{\uparrow} = \mathbb{I}_A \neq \amalg_{AA}$,
2. $(R^{\downarrow})^{\uparrow} = R^{\downarrow}$,
3. $(R^{\uparrow})^{\downarrow} = R^{\uparrow}$,
4. $^{\uparrow}$ is a closure[10] and $^{\downarrow}$ is a kernel operation,
5. R = $R^{\uparrow}$ iff $R^{\downarrow} = R^{\uparrow}$ iff $R^{\downarrow}$ = R,
6. $\amalg_{AB}^{\uparrow} = \amalg_{AB}$ and $\pi_{AB}^{\downarrow} = \pi_{AB}$,
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. $(Q \sqcap R^{\uparrow})^{\uparrow} = Q^{\uparrow} \sqcap R^{\uparrow}$,
12. $Q^{\uparrow} \setminus T^{\downarrow} = (Q^{\uparrow} \setminus T)^{\downarrow} \sqsubseteq (Q \setminus T)^{\downarrow}$ and $(Q \setminus T)^{\uparrow} \sqsubseteq Q^{\downarrow} \setminus T^{\uparrow}$,
13. $T^{\downarrow} / S^{\uparrow} = (T/S^{\uparrow})^{\downarrow} \sqsubseteq (T/S)^{\downarrow}$ and $(T/S)^{\uparrow} \sqsubseteq T^{\uparrow} / S^{\downarrow}$.

Closure Properties of the Class of Crisp Relations in Arrow Category In the next lemma, some closure properties of the class of crisp relations in Arrow categories are collected [61]:

Lemma 2.4.8. Let $\mathcal{A}$ be an arrow category and Q_i, Q: A → B for i ∈ I, R: A → C, and S: B → C be crisp relations. Then the following holds:

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 \ R are crisp.

[10] Also called as *support*.

Symmetric Quotient Property in Arrow Category Symmetric quotients in arrow categories is considered as one of the most important properties in this category.

Definition 2.4.13. Let $\mathcal{A}$ be an arrow category with Q: A $\to$ B and R: C $\to$ B to be crisp relations. if $syq(Q,R)^{\downarrow}$ is surjective, then the following holds [61]:

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

2.4.4.3 Boolean Arrow category

Definition 2.4.14. An arrow category $\mathcal{A}$ is said to be a *Boolean arrow category* if there is a Boolean algebra $\mathcal{B}$ and a mapping F: $\mathcal{A} \to \mathcal{B}$ from objects of $\mathcal{A}$ to elements of $\mathcal{B}$, such that for all objects A and B in $\mathcal{A}$, we have F(A) $\leq$ F(B) in $\mathcal{B}$ iff there is an arrow f: A $\to$ B in $\mathcal{A}$ [2].

Lemma 2.4.9. Let $\mathcal{A}$ be a Boolean arrow category and Q, R: A $\to$ B. Then we have [61, 62]:

1. $\overline{Q^{\downarrow}} = \overline{Q}^{\uparrow}$ and $\overline{Q^{\uparrow}} = \overline{Q}^{\downarrow}$,
2. $Q^{\uparrow} \sqcup R^{\downarrow} = (Q^{\uparrow} \sqcup R)^{\downarrow}$,
3. $Q^{\uparrow} \to R^{\downarrow} = (Q \to R^{\downarrow})^{\downarrow}$ and $(Q \to R)^{\uparrow} = Q^{\downarrow} \to R^{\uparrow}$.

Proofs of the above properties in boolean arrow categories can be found in [61].

2.4.5 Fuzzy Category

Fuzzy Category $\mathcal{F}$ is an *Arrow* category extended by *-based (* , $\overset{*}{;}$ and $\twoheadrightarrow$) operations [1, 65].

1. if Q, R: A $\to$ B:
   - Q $*$ R: A $\to$ B,
   - Q $*$ $R^{\downarrow}$ = Q $\sqcap$ $R^{\downarrow}$,
   - $(Q * R)^{\smile} = Q^{\smile} * R^{\smile}$.
2. if Q: A $\to$ B and R: B $\to$ C:
   - Q $\overset{*}{;}$ R: A $\to$ C,

- Q ⨾* $R^{\downarrow}$ = Q; $R^{\downarrow}$,
- (Q ⨾* R)$^{\smile}$ = $R^{\smile}$ ⨾* $Q^{\smile}$.

3. if R : B → C and S: A → C, and X: A → B:
   - S ⫽* R: A → B,
   - X ⨾* R ⊑ S ⇔ X ⊑ S ⫽* R.

$\mathcal{L}$-REL is an example for all *Heyting*, *Arrow*, and *Fuzzy* categories. In REL (regular/Boolean/crisp relations), the arrow operations are *identity* and $*$ to be *meet*.
This means that any theory developed in a fuzzy category applies to both situations.

2.4.6 Relational Powers

The next construction we are interested in is an internal version of a power set, i.e., the object of all subsets of a given object. As already mentioned, in the case of $\mathcal{L}$-fuzziness, we are interested in $\mathcal{L}$-fuzzy subsets, of course [1].

Definition 2.4.15. An object P(A) together with a relation ϵ: A → P(A) is called a relational power iff

1. $syq(\epsilon, \epsilon)^{\downarrow} = \mathbb{I}_{P(A)}$,
2. $syq(R, \epsilon)^{\downarrow}$ is *total* for every R: A → B.

we will use the abbreviation $\Lambda(R) = syq(R^{\smile}, \epsilon)^{\downarrow}$ [1].

Now, we obtain the following lemma. A proof can be found in [1].

Lemma 2.4.10. Let L^B and L^C be a relational power and R : A → B and S: L^B → C be a relations. Then we have:

- $\Lambda(R)$ is a map.
- $\Lambda(R)$; $\epsilon^{\smile}$ = R.
- $\Lambda(R)$; $\Lambda(S)$ = $\Lambda(\Lambda(R); S)$.

Also, we can define $\Omega = (\epsilon \setminus \epsilon)^{\downarrow}$ so that we have Ω (M, N) = 1 iff M(a) ⩽ N(a) for all a ∈ A [1].

2.4.7 Relational Product

Relational product as a type of relational construction in Dedekind categories is discussed in this section. So far, we represented these properties in Lemma. 2.2.2. Now, we look them as an abstract definition of relational product. This definition is provided in the context of arrow categories and requires the projections to be crisp. The definition of relational product can be found in [61]:

Definition 2.4.16. An object $A \times B$ together with two crisp relations π: $A \times B \to A$ and ρ: $A \times B \to B$ is called a *relational product* iff the following holds [61]:

$$\pi^{\smile} ; \pi \sqsubseteq \mathbb{I}_A, \quad \rho^{\smile} ; \rho \sqsubseteq \mathbb{I}_B, \quad \pi^{\smile} ; \rho = \pi_{AB}, \quad \pi^{\smile} ; \pi \sqcap \rho^{\smile} ; \rho = \mathbb{I}_{A \times B}$$

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

Chapter 3

Formal Concept Analysis

Formal Concept Analysis (Ganter et al., 1999; Wille [57], 1982) is a mathematical framework that underlies many methods of knowledge discovery and data analysis. Over the past 30 years the initial theory has been combined with and enriched by other research domains in mathematics including description logics and conceptual graphs [50]. In this chapter, two fundamental definitions including Formal Concept Analysis (FCA) and Fuzzy Formal Concept Analysis (FFCA) are defined along with some related disciplines.

3.1 Formal Concept Analysis (FCA) and related disciplines

Formal concept analysis (*FCA*), achieved aims using some formulations in the German standards on concepts and conceptual systems; these standards are seen as a general aid in sciences, economy, and administration for a better understanding and use of ”conceptual tools.” The standards are based on the philosophical understanding of a concept as a unit of thoughts consisting of two parts: the extension and the intension (comprehension); the extension covers all *objects* (or *entities*) belonging to the concept while the intension comprises all *attributes* (or *properties*) valid for all those objects. A set-theoretic model for these relationships is the root of formal concept analysis. This model yields not only an approach to data analysis, but also methods for formal representation of conceptual knowledge [58].

3.1.1 Overview of the FCA Approach

Formal concept analysis starts with the primitive notion of a *formal context* and then continues with the notion of *formal concept*. Finally, concept Lattice of a formal context is generated to illustrate the relations. These structures constitute the basis of formal concept

analysis. This is a lattice-based method for the analysis of hierarchies of concepts and has especially proved to be very useful in many computer science applications, for instance, in knowledge representation and discovery, information retrieval, data mining, and program analysis [10].

To put it more simply, in FCA, data is represented as a conceptual hierarchy, organized as a concept lattice that relates objects and their properties [46]. In other words, two notions can be derived by FCA: formalisation of concepts and conceptual thinking and also, how objects can be hierarchically studied together to identify groups of elements or objects with common sets of attributes as a well-established technique [4].

The processing steps are illustrated in Fig. 3.1 [4]. The initial step to generate FCA is the source code or input data. Then, formal contexts and concepts are made by finding the relations among all objects of input data using FCA mapping. The concept lattices can be formed as the final step in the FCA overall approach as shown in Fig. 3.1.

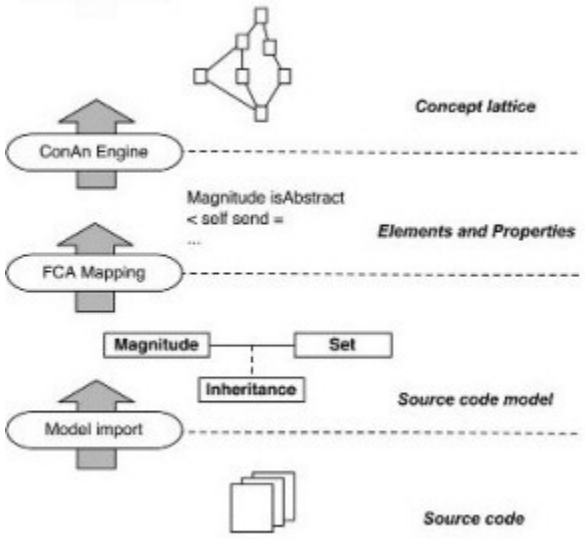


Figure 3.1: The overall approach of FCA.

In the next section, formal context is discussed including some examples to clarify it comprehensively.

3.1.2 Formal Context in FCA

To analyze the hierarchies of concepts, *formal context* is the first step, which consists of a non-empty set of *objects*, a non-empty set of *attributes* and a *relation* with the objects as source and the attributes as target [10].

Definition 3.1.1. A *formal context* is a triple $\mathbb{K}$ = **(G, M, I)**, where G and M are sets, **I** $\subseteq$ **G** $\times$ **M** is a binary relation from G to M. In a formal context (G, M, I), G is interpreted as the set of *objects*, M the set of *attributes*, and **(G, M)** $\in$ **I** reads as that the object G has property M [10, 45].

In the literature on the *formal concept analysis* (e.g., in [33, 34]), the relation frequently is denoted with the letter **I**, called the *incidence relation*, and typed with **G** (from the German word "**G**egenstande") as *source* and **M** (from the German word "**M**erkmale") as *target* [1] [10].
Let us illustrate the notion of concept of a formal context using the data in "bodies of water" [2] table again, which is a very well-known real example. The data in the example is taken from a semantic field study, where different kinds of bodies of water were systematically categorized by their attributes. For the purpose here it has been simplified.

Example 3.1.1. In the following table, the data table represents a formal context.

$$\mathbf{G}_{objects} = \{ \text{canal, channel, lagoon, lake, maar, puddle, pond, pool, reservoir, river, rivulet, runnel, sea, stream, tarn, torrent, trickle} \}$$

$$\mathbf{M}_{attributes} = \{ \text{temporary, running, natural, stagnant, constant, maritime} \}$$

$$\mathbf{I} = \{ \langle \text{canal, running} \rangle, \langle \text{Lagoon, maritime} \rangle, \langle \text{maar, natural} \rangle, \langle \text{puddle, temporary} \rangle, \langle \text{tarn, constant} \rangle \ldots \}$$

Considering Fig. 3.2, relation I is denoted a subset of G *Cartesian product* M or simply, a subset of pairs.
To indicate that a specific object has a specific attribute or not, the table or matrix contents are filled by **1**, **0** or ***yes***, ***no*** respectively. Since there are some binary values, I can be called as a *binary relation* in FCA.

3.1.2.1 Derivation operators or mappings in Formal Context

The way FCA looks at concepts is that it formulates the following definition: "A concept is considered to be a unit of thought constituted of two parts: its *extent* and its *intent*." The ***extent*** consists of all ***objects*** belonging to the concept, while the ***intent*** comprises all ***attributes*** **shared by those objects** [50].

[1] For representing incidence relations frequently so-called cross-tables are used. These are nothing else than specific representations of Boolean matrices or database tables.

[2] `https:\\en.wikipedia.org\wiki\Formal_concept_analysis`

bodies of water		attributes					
		temporary	*running*	*natural*	*stagnant*	*constant*	*maritime*
objects	canal		✓			✓	
	channel		✓			✓	
	lagoon			✓	✓	✓	✓
	lake			✓	✓	✓	
	maar			✓	✓	✓	
	puddle	✓		✓	✓		
	pond			✓	✓	✓	
	pool			✓	✓	✓	
	reservoir				✓	✓	
	river		✓	✓		✓	
	rivulet		✓	✓		✓	
	runnel		✓	✓		✓	
	sea			✓	✓	✓	✓
	stream		✓	✓		✓	
	tarn			✓	✓	✓	
	torrent		✓	✓		✓	
	trickle		✓	✓		✓	

Figure 3.2: Example for a *formal context* - **Bodies of Water**.

Definition 3.1.2. The operations $\triangle$: $\mathcal{P}(G) \rightarrow \mathcal{P}(M)$ and $\triangledown$: $\mathcal{P}(M) \rightarrow \mathcal{P}(G)$ are defined by [10, 45, 50, 58]:

1. For a set of objects O $\subseteq$ G, the set of **common attributes** can be defined by:

$$O^{\triangle\ 3} = \{\ m \in M \mid (g, m) \in I \text{ for all } g \in O\ \}$$

2. For a set of objects A $\subseteq$ M, the set of **common objects** can be defined by:

$$A^{\triangledown\ 4} = \{\ g \in G \mid (g, m) \in I \text{ for all } m \in A\ \}$$

In $\triangle$ and $\triangledown$ definitions, $\mathcal{P}(G)$ shows *power set of objects* and $\mathcal{P}(M)$ is *power set of attributes*.

[3] Upper bounds in the case of ordered sets [10].
[4] Lower bounds in the case of ordered sets [10].

Let us illustrate the notion of concept of a formal context using the data in Fig. 3.2.

Example 3.1.2. Using Example. 3.1.1 we get:

$$\{lagoon, lake, maar, pond, pool, sea, tarn\}^{\triangle} = \{\text{natural, stagnant, constant}\}$$

$$\{running, natural, constant\}^{\triangledown} = \{\text{river, rivulet, runnel, stream, torrent, trickle}\}$$

3.1.3 Formal Concept in FCA

Pair **(O, A)** is a *formal concept*:

(i) Every object in O has every attribute in A.

(ii) For every object in G that is not in O, there are some attributes in A that the mentioned object does not have those attributes.

(iii) For every attribute in M that is not in A, there are some objects in O that do not have that attribute.

Precisely, the definition of *formal concept* is described as below:

Definition 3.1.3. pair **(O, A)** is a *formal concept* of a *context* **(G, M, I)** if [10, 50, 58]:

$$\text{O} \subseteq \text{G}, \quad \text{A} \subseteq \text{M}, \quad O^{\triangle} = \text{A}, \quad A^{\triangledown} = \text{O}.$$

Based on the relation I: G $\rightarrow$ M, a formal concept then is a pair (O, A) from $\mathcal{P}(G) \times \mathcal{P}(M)$ such that the equations $O^{\triangle} = \text{A}$ and $A^{\triangledown} = \text{O}$ hold [10].

Example 3.1.3. According to Example. 3.1.2 we have:

$$\Big\{\text{lagoon, lake, maar, pond, pool, sea, tarn}\Big\}^{\triangle} = \Big\{\text{natural, stagnant, constant}\Big\}$$

and also,

$$\Big\{\text{natural, stagnant, constant}\Big\}^{\triangledown} = \Big\{\text{lagoon, lake, maar, pond, pool, sea, tarn}\Big\}$$

So that the following pair is a *formal concept*:

$$\Big(\{\text{lagoon, lake, maar, pond, pool, sea, tarn}\}, \quad \{\text{natural, stagnant, constant}\}\Big)$$

3.1.4 Concept Lattice of a Formal Context in FCA

By trying to join formal concepts in formal concept analysis, a concept lattice can be produced as follows [12, 10, 58]

Definition 3.1.4. If (O_1, A_1) and (O_2, A_2) are two formal concepts, then (O_1, A_1) is defined to be less general or equal than (O_2, A_2), denoted by $(O_1, A_1) \sqsubseteq (O_2, A_2)$, if $O_1 \subseteq O_2$ or, equivalently, $A_1 \supseteq A_2$. With this relation the set

$$\mathcal{B}(I) := \{(\mathrm{O}, \mathrm{A}) \in 2^G \times 2^M \mid O^{\triangle} = \mathrm{A} \quad \wedge \quad A^{\triangledown} = \mathrm{O}\}$$

of all concepts constitutes a *complete lattice* ($\mathcal{B}(I), \sqsubseteq$), in [34] called *concept lattice* ([7] speaks of a *Galois lattice*).

In other words, there is a natural hierarchical ordering relation between the concepts of a given context that is called the *subconcept-superconcept relation* [50].
For any two concepts (O_1, A_1) and (O_2, A_2):

$$(\, O_1, A_1 \,) \leq (\, O_2, A_2 \,) \iff O_1 \subseteq O_2 \iff A_2 \subseteq A_1.$$

A concept $C_1 = (O_1, A_1)$ is called a *subconcept* of a concept $C_2 = (O_2, A_2)$ (or equivalently, C_2 is called a *superconcept* of a concept C_1) if the extent of C_1 is a subset of the extent of C_2 (or equivalently, if the intent of C_1 is a superset of the intent of C_2).

Example 3.1.4. Concept Lattice corresponding to the formal context Bodies of Water in Fig. 3.2:
Table 3.2 can be derived from the line diagram in Fig. 3.3 by applying the following reading rule: An object is described by an attribute if and only if there is an ***ascending path*** from the object to the attribute (attributes). Objects are located below and attributes are located above concept circles. By way of illustration, to **retrieve the *intent* of a formal concept** one traces all paths leading up from the corresponding node in order to collect all attributes [50].
The ***extent*** of a concept consists of those objects from which an ascending path leads to the circle representing the concept. The ***intent*** consists of those attributes to which there is an ascending path from that concept circle (in the diagram of the Fig. 3.3). In this diagram **the concept immediately to the left of the label *reservoir*** (which is unnamed) has the **intent** *stagnant* and *natural* and the **extent** *puddle*, *maar*, *lake*, *pond*, *tarn*, *pool*, *lagoon*, and *sea*. A concept is a subconcept of all concepts that can be reached by travelling upward. This concept will inherit all attributes associated with these superconcepts [50].

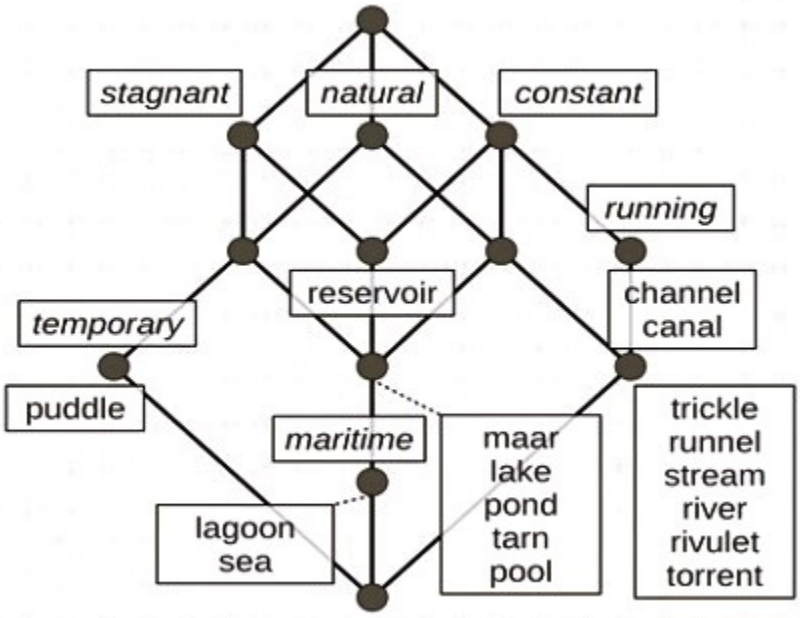


Figure 3.3: Concept Lattice corresponding to the formal context ”Bodies of Water”.

Given the following example, We clarify these concepts:

Example 3.1.5. A labelled line diagram of the concept lattice of the context given by Fig. 3.4 is shown in Fig. 3.5. The little circles represent the 65 concepts of the context and the ascending paths of line segments represent the subconcept-superconcept-relation (such a path may change its direction at a meeting of lines only if there is a little circle or a dot) [58]. As an example, by following the red arrows from Charlotte (object) in ascending paths, four code numbers (3, 4, 5, and 7) of social events (attributes) can be read in Fig. 3.5.

3.2 Fuzzy Formal Concept Analysis (FFCA) and related disciplines

To prepare a brief summary, fuzzy formal concept analysis is an extension of the original formal concept analysis. In fuzzy formal concept analysis, degrees are taken from a scale $\mathcal{L}$ instead of Boolean values .

	Code Numbers of Social Events Reported in Old City Herald													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Evelyn	x	x	x	x	x	x		x	x					
Laura	x	x	x		x	x	x	x						
Theresa		x	x	x	x	x	x	x	x					
Brenda	x		x	x	x	x	x	x						
Charlotte			x	x	x		x							
Frances			x		x	x		x						
Eleanor					x	x	x	x						
Pearl						x		x	x					
Ruth					x		x	x	x					
Verne							x	x	x			x		
Myra								x	x	x		x		
Katherine								x	x	x		x	x	x
Sylvia							x	x	x	x		x	x	x
Nora						x	x		x	x	x	x	x	x
Helen							x	x		x	x	x		
Dorothy								x	x					
Olivia									x		x			
Flora									x		x			

Figure 3.4: Participation of social events by some ladies in Old City.

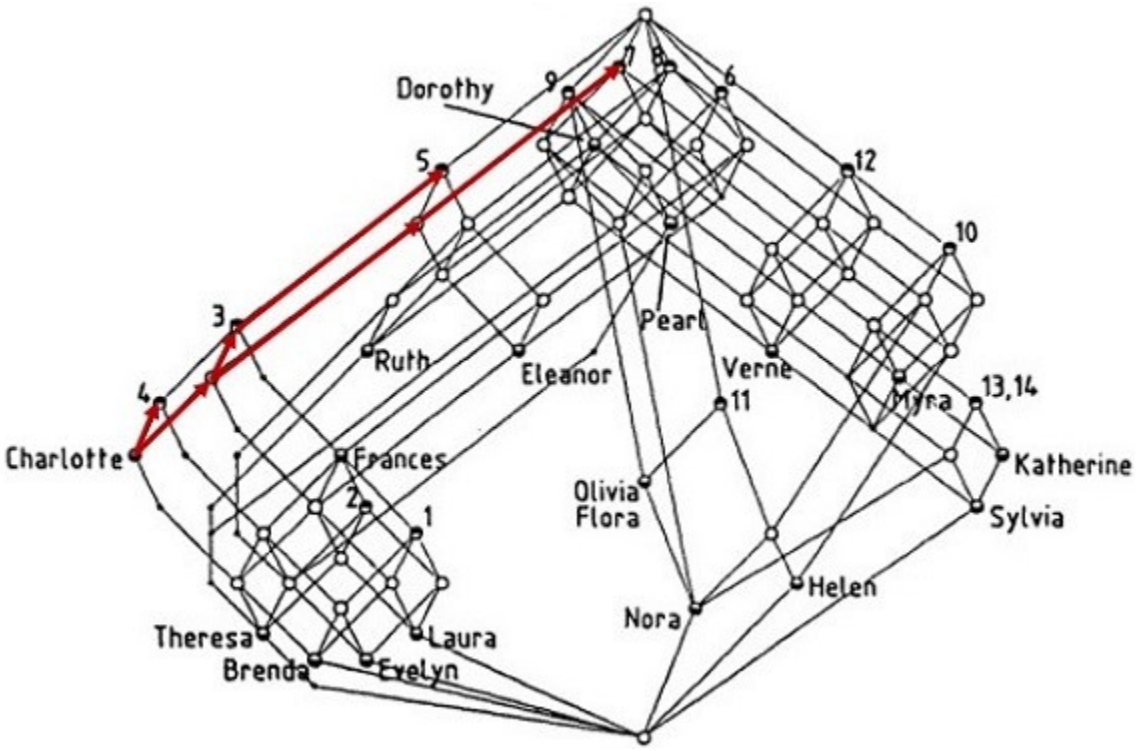


Figure 3.5: Concept lattice of the formal context in Fig. 3.4.

3.2.1 Overview of the Fuzzy FCA (FFCA) Approach

Besides the ordinary set, fuzzy set theory permits uncertainty information that is directly represented by membership value in the range of [0, 1]. The membership value indicates the grade of membership of set elements. If an element is mapped to the value 0, the element is not included in the fuzzy set, and 1 describes a fully included element. In order to analyze vague data set of uncertainty information, Fuzzy Formal Concept Analysis (Fuzzy FCA) incorporates fuzzy set theory into FCA. It extracts useful information with a unit of fuzzy concept from given fuzzy formal context with a confidence threshold, and constructs fuzzy lattice by order relations between the fuzzy concepts [56, 67, 68].

3.2.2 Fuzzy Formal Context in FFCA

A *fuzzy formal context* is a triple $\mathbb{K} := (G, M, I_f)$ where G is a finite set of *objects*, M is a finite set of *attributes*, and I_f is a *fuzzy set* on domain $\mathbf{G} \times \mathbf{M}$.
Each relation (g, m) $\in I_f$ has a membership value in [0, 1] [67, 68].

Simply in FFCA, I_f**:** $\mathbf{G} \times \mathbf{M} \longrightarrow \mathcal{L}$ means that a *fuzzy binary relation* assigns a degree I_f**(g, m)** $\in \mathcal{L}$ to each g $\in$ G and each m $\in$ M to show that *object* g has *attribute* m with *degree* I_f.
A *fuzzy formal context* can also be represented as a *cross-table* as presented in Table. 3.1. The context has objects as G = { O_1, O_2, O_3, O_4 }. It also has attributes as M = { a, b, c, d }. Each relation between objects and attributes is represented by a membership value [68].

Table 3.1: Fuzzy Formal Context.

Attribute List				
	a	**b**	**c**	**d**
O_1	0.2	0.1	0.9	0.9
O_2	1.0	0.4	0.3	0.1
O_3	0.0	0.4	0.2	0.1
O_4	0.2	0.1	0.7	0.0

3.2.2.1 Confidence Threshold $\mathbb{T}$ in Fuzzy FCA (FFCA)

A *confidence threshold* $\mathbb{T}$ has an interval $[t_1, t_2]$, where $0 \leq t_1 < t_2 \leq 1$. By using the confidence threshold $\mathbb{T}$, we can eliminate some relations that are out of the interval values from a given fuzzy context. The confidence threshold $\mathbb{T}$ can be set by user according to

the application or the domain knowledge. For instance, Table. 3.3 shows a *fuzzy formal context* with $\mathbb{T}$ = [0.3, 1.0] [67, 68].

Table 3.2: Fuzzy formal context with $\mathbb{T}$ = [0.3, 1.0].

	Attribute List			
	a	**b**	**c**	**d**
O_1	–	–	0.9	0.9
O_2	1.0	0.4	0.3	–
O_3	–	0.4	–	–
O_4	–	–	0.7	–

3.2.2.2 Derivation operators or mappings in Fuzzy Formal Context

Definition 3.2.1. Given two fuzzy sets O ⊆ G of objects and a fuzzy set A ⊆ M of attributes, two operator $^{\triangle}$ and $^{\triangledown}$ are defined respectively:

1. $O^{\triangle}(\mathrm{m}) = \bigwedge_{g \in G} \Big(\mathrm{O(g)} \longrightarrow I_f(g, m) \Big)$
2. $A^{\triangledown}(\mathrm{g}) = \bigwedge_{m \in M} \Big(\mathrm{A(m)} \longrightarrow I_f(g, m) \Big)$

3.2.3 Fuzzy Formal Concept in FFCA

Definition 3.2.2. Similar to Formal Concept Analysis (FCA), pair (O, A) ∈ G × M is called a *fuzzy formal concept* if $O^{\triangle}$ = A and $A^{\triangledown}$ = O for O ⊆ G and A ⊆ M. The fuzzy sets O and A are called the *extent* and *intent* of the concept, respectively [56, 67, 68].

Example 3.2.1. The following table indicates fuzzy formal concepts of Table. 3.2 [68].

Table 3.3: Fuzzy formal context with $\mathbb{T}$ = [0.3, 1.0].

	Extents	**Intents**
C_1	{ }	{a(1.0), b(1.0), c(1.0), d(1.0)}
C_2	$\{O_2(0.3)\}$	{a(1.0), b(0.4), c(0.3)}
C_3	$\{O_1(0.9)\}$	{c(0.9), d(0.9)}
C_4	$\{O_2(0.4), O_3(0.4)\}$	{ b(0.4) }
C_5	$\{O_1(0.9), O_2(0.3), O_4(0.7)\}$	{ c(0.7) }
C_6	$\{O_1(1.0), O_2(1.0), O_3(1.0), O_4(1.0)\}$	{ }

Example 3.2.2. Let $\mathbb{K}$ = (X, Y, R) be an $\mathcal{L}$-fuzzy context where X = { x_l, x_2, x_3 } is a *symptom* set, Y = { y_l, y_2, y_3 } an *illness* set, $\mathcal{L}$ = { 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1 } and the $\mathcal{L}$-fuzzy relation defined as follows [17]:

Table 3.4: Symptom-Illness $\mathcal{L}$-Fuzzy Relations with $\mathcal{L}$ = [0, 1].

symptom List			
	y_1	y_2	y_3
x_1	0.6	0.6	0
x_2	0.9	0.5	0.3
x_3	1	0.2	0.7

that represents the relationship between the illnesses and the symptoms. Applying the *up* and *down* triangles and generating all fuzzy concepts, we construct the $\mathcal{L}$-fuzzy concept lattice in the next section formed by following $\mathcal{L}$-fuzzy concepts.

$A_1 = \{ x_1(0.3), x_2(0.3), x_3(0.3) \}$,

$B_1 = \{ y_1(0.7), y_2(0.7), y_3(0.7) \}$,

$A_2 = \{ x_1(0.3), x_2(0.3), x_3(0.4) \}$,

$B_2 = \{ y_1(0.7), y_2(0.6), y_3(0.7) \}$,

$A_3 = \{ x_1(\mathrm{O}.3), x_2(\mathrm{O}.3), x_3(0.5) \}$,

$B_3 = \{ y_1(0.7), y_2(0.5), y_3(0.7) \}$,

$A_4 = \{ x_1(\mathrm{O}.3), x_2(\mathrm{O}.3), x_3(0.6) \}$,

$B_4 = \{ y_1(0.7), y_2(0.4), y_3(0.7) \}$,

$A_5 = \{ x_1(\mathrm{O}.3), x_2(\mathrm{O}.3), x_3(0.7) \}$,

$B_5 = \{ y_1(0.7), y_2(0.3), y_3(0.7) \}$,

$A_6 = \{ x_1(0.4), x_2(0.4), x_3(0.4) \}$,

$B_6 = \{ y_1(0.6), y_2(0.6), y_3(0.6) \},$

$A_7 = \{ x_1(\mathrm{O}.4), x_2(\mathrm{O}.4), x_3(0.5) \},$

$B_7 = \{ y_1(0.6), y_2(0.5), y_3(0.6) \},$

$A_8 = \{ x_1(\mathrm{O}.4), x_2(\mathrm{O}.4), x_3(0.6) \},$

$B_8 = \{ y_1(0.6), y_2(0.4), y_3(0.6) \},$

$A_9 = \{ x_1(0.4), x_2(0.4), x_3(0.7) \},$

$B_9 = \{ y_1(0.6), y_2(0.3), y_3(0.6) \},$

$A_{10} = \{ x_1(0.5), x_2(0.5), x_3(0.5) \},$

$B_{10} = \{ y_1(0.6), y_2(0.5), y_3(0.5) \},$

$A_{11} = \{ x_1(\mathrm{O}.5), x_2(\mathrm{O}.5), x_3(0.6) \},$

$B_{11} = \{ y_1(0.6), y_2(0.4), y_3(0.5) \},$

$A_{12} = \{ x_1(\mathrm{O}.5), x_2(\mathrm{O}.5), x_3(0.7) \},$

$B_{12} = \{ y_1(0.6), y_2(0.3), y_3(0.5) \},$

$A_{13} = \{ x_1(0.6), x_2(0.5), x_3(0.5) \},$

$B_{13} = \{ y_1(0.6), y_2(0.5), y_3(0.4) \},$

$A_{14} = \{ x_1(\mathrm{O}.6), x_2(\mathrm{O}.6), x_3(0.6) \},$

$B_{14} = \{ y_1(0.6), y_2(0.4), y_3(0.4) \},$

$A_{15} = \{ x_1(\mathrm{O}.6), x_2(\mathrm{O}.6), x_3(0.7) \},$

$B_{15} = \left\{ y_1(0.6), y_2(0.3), y_3(0.4) \right\}$,

$A_{16} = \left\{ x_1(\text{O}.6), x_2(\text{O}.7), x_3(0.7) \right\}$,

$B_{16} = \left\{ y_1(0.6), y_2(0.3), y_3(0.3) \right\}$.

For every $\mathcal{L}$-fuzzy concept, the first and the second lines represent the first and second elements of the pair, respectively. Moreover, to interpret an $\mathcal{L}$-fuzzy concept we look at the membership of the objects and attributes which stand out from the others. For example, if we look at the $\mathcal{L}$-fuzzy concept (A_8, B_8), we can see that three illnesses y_1, y_2, and y_3 have the degrees 0.6, 0.4, and 0.6, respectively. Likewise, this concept has three symptoms x_1, x_2, and x_3 with degrees 0.4, 0.4, and 0.6, respectively. Then we can say that the patients having the illnesses y_l and y_3 have mainly the symptom x_3, and this symptom appears in a least level in the illness y_2.

It is necessary to note that a concept is only given by sets (A, B) that satisfy $A^{\triangle}$ = B and $B^{\triangledown}$ = A. For table. 3.4, in total we have 11 $^{\wedge}$ 3 = 1,331 subsets of G (objects) and 11 $^{\wedge}$ 3 = 1,331 subsets of M (attributes). This gives 1,331 * 1,331 = 1,771,561 pairs of such sets. Only a few of those pairs are concepts.

3.2.4 Concept Lattice of a Fuzzy Formal Context in FFCA

Applying concepts of $O^{\triangle}$(m) and $A^{\triangledown}$(g) and using the set of all formal fuzzy concepts, we can construct the $\mathcal{L}$-fuzzy concept lattice formed by the $\mathcal{L}$-fuzzy concepts.

Example 3.2.3. Considering all concepts in Example. 3.2.2, the related lattice is generated in Fig. 3.6 [17].

The circles represent the $\mathcal{L}$-fuzzy concepts. We take the *objects* and *attributes* of the context the *join-irreducible* and *meet-irreducible* elements of $\mathcal{L}$, respectively. We see that the subset of *join-irreducible elements* of the lattice {2, 3, 4, 5, 6, 10, 13, 16} is represented by *shaded lower half-circles* and the *meet-irreducible elements* {1, 5, 6, 9, 12, 13, 14, 15}, by *shaded upper half-circles*. The $\mathcal{L}$-fuzzy concepts 5, 6 and 13 are both *join and meet-irreducible elements*.

In the graph of Fig. 3.6, we also look at the hierarchy of concepts through the lines

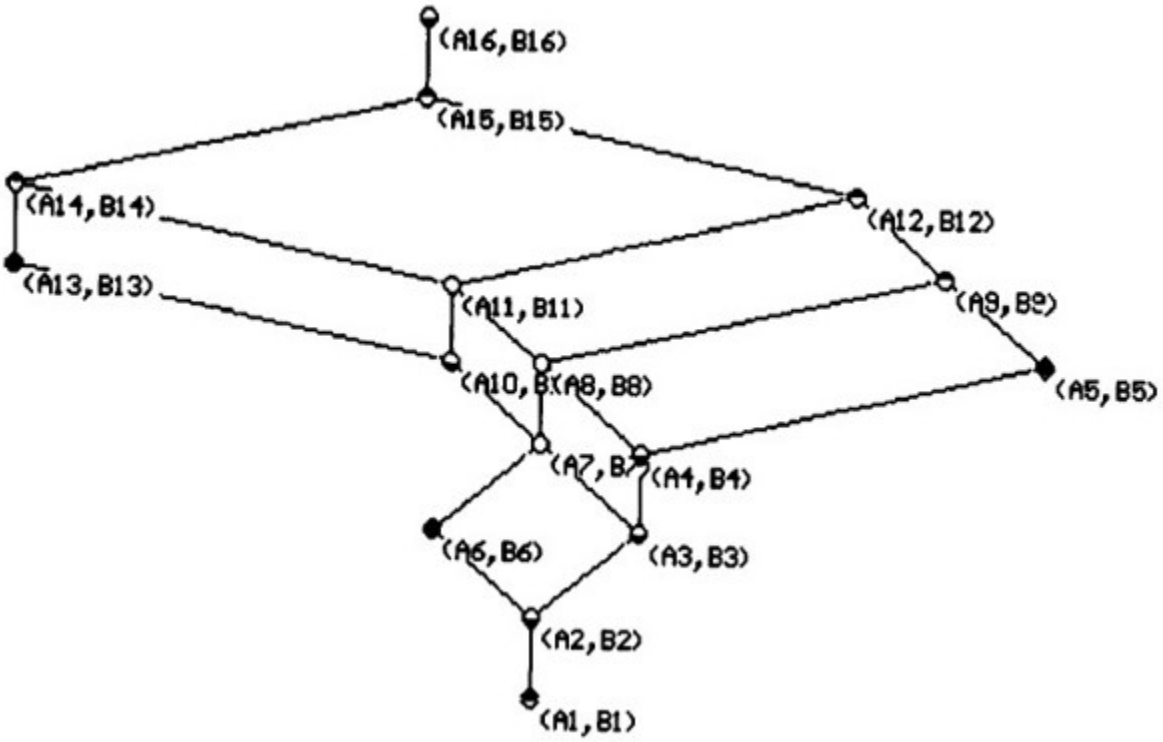


Figure 3.6: $\mathcal{L}$-fuzzy concept lattice formed by $\mathcal{L}$-fuzzy concepts in example of symptom-illness.

between them. In simple words, the graph shows the order between concepts. If there is a line between two concepts, then the one which is drawn lower is smaller than the one which is drawn higher. Such a graph is called a *Hasse diagram* of an ordered set. For example, the concepts (A_{11}, B_{11}) and (A_{13}, B_{13}) are in no order relationship, i.e., neither (A_{11}, B_{11}) is smaller than (A_{13}, B_{13}) nor vice versa.

Definition 3.2.3. Similar to Concept lattice in FCA, we have the following hierarchical ordering in FFCA:

$$(X_1, Y_1) \leq (X_2, Y_2) \text{ when } X_1 \subseteq X_2 \quad \text{or} \quad Y_1 \supseteq Y_2.$$

Example 3.2.4. Fig. 3.7 illustrates fuzzy concept lattice of the concepts in Table. 3.3.

As an example, considering (A_{14}, B_{14}) and (A_{15}, B_{15}) in Example. 3.2.2, $A_{14} \subseteq A_{15}$ or $B_{15} \subseteq B_{14}$:

$$x_1/\text{O}.6 \leq x_1/\text{O}.6 \quad \text{and} \quad x_2/\text{O}.6 \leq x_2/\text{O}.6 \quad \text{and} \quad x_3/0.6 \leq x_3/0.7$$

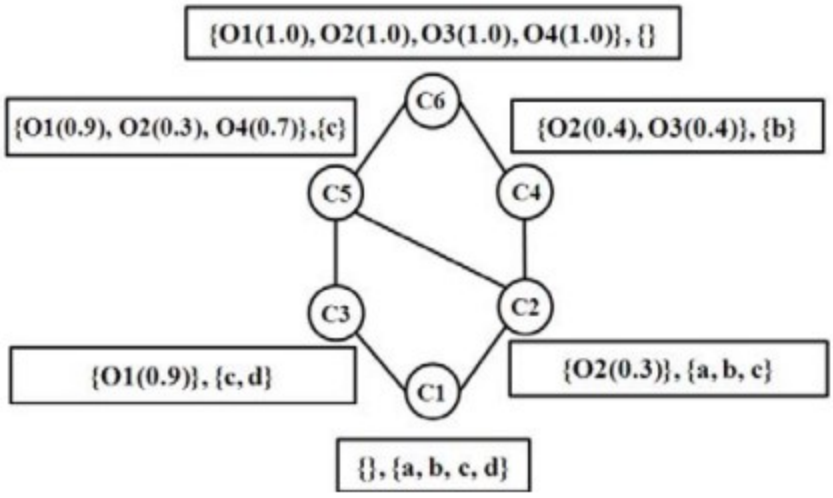


Figure 3.7: Fuzzy Concept Lattice.

OR

$y_1/0.6 \leq y_1/0.6$ and $y_2/0.3 \leq y_2/0.4$ and $y_3/0.4 \leq y_3/0.4$

For fuzzy sets, the term of *subset* just means that the degree of each element in the smaller set is *less* or *equal* to the degree of that element in the bigger set.

Chapter 4

Relational Approach

4.1 Relational Concept Analysis

So far, working on two types of boolean and fuzzy forms of the relations has been the core of our focus in this thesis. Also, we figured out formal and fuzzy formal contexts and subsequently, we derived concepts to be able to generate the related lattices. Now, we look at all those definitions relation-algebraically getting through adding some arrows and creating some special changes in the former formulas.

Generally, this chapter is dealing with a relational style of the concept analysis. Accomplishing this goal is totally associated with presenting new definitions of *Up* and *Down* triangles ($^{\triangle}$ and $^{\triangledown}$ respectively) using a relational point of view.

Generally, we want to show in the case of $\mathcal{L}$-fuzzy relations and subsets that a formal concept is completely determined by its extent (or by its intent), i.e., it is determined by its set of object (or by its set of attributes). First, we define partial identities, which characterize formal concepts as a pair, the object set of a formal concept (called extents), and the attribute set of a formal concept (called intents) [1]. In other words, it is suggested to consider a set of objects or a set of attributes rather than pairs of them dealing with formal concepts. This is the key behind the definition of *partial identity*. We also take advantage of visual representations of the new relational approach for better understanding.

We will phrase our approach in terms of abstract arrow resp. fuzzy categories as introduced in Section. 2.4. Since regular relations establish a special case of fuzzy categories, we cover fuzzy as well as regular concept analysis within the same framework.

Definition 4.1.1. Let I be a relation such that I: G $\longrightarrow$ M, i.e., a relation between *objects* and *attributes*, $\triangle$: P_L(G) $\longrightarrow$ P_L(M), and ϵ[1]: G $\longrightarrow$ 2^G ($M \longrightarrow 2^M$). Then we have:

$$\triangle \;=\; syq\Big((\epsilon \setminus I)^{\smile}, \epsilon\Big)^{\downarrow} = syq\Big(I^{\smile}/\epsilon^{\smile}, \epsilon\Big)^{\downarrow}$$

Definition 4.1.2. Let O $\in$ G, A $\in$ M, I be a relation such I: G $\longrightarrow$ M, i.e., a relation between *objects* and *attributes*, $\triangledown$: P_L(M) $\longrightarrow$ P_L(G), and ϵ: G $\longrightarrow$ 2^G ($M \longrightarrow 2^M$). Then we have:

$$\triangledown \;=\; syq\Big((\epsilon \setminus I^{\smile})^{\smile}, \epsilon\Big)^{\downarrow} = syq\Big(I/\epsilon^{\smile}, \epsilon\Big)^{\downarrow}$$

$\triangle$ maps sets of objects to the sets of attributes and conversely, $\triangledown$ maps sets of attributes to the sets of objects. There can be a confusion here between a *relation* I and the *identity relation* $\mathbb{I}$. I here is not $\mathbb{I}$, it is an arbitrary relation between G and M.
Accordingly, the composition of the two above functions is defined as below [10]

Lemma 4.1.1. Consider I: G $\longrightarrow$ M be a relation and ϵ: G $\longrightarrow$ 2^G ($M \longrightarrow 2^M$), then we can derive:

$$\triangle \,;\, \triangledown = \text{syq}\left(\text{ I / } (\epsilon \setminus \text{I}),\, \epsilon \right)^{\downarrow}$$

Proof 4.1.1. Now, we indicate the proof of above equation ($\triangle$; $\triangledown$) step by step:

$$\begin{aligned}
\triangle;\triangledown &= syq\Big((\epsilon \setminus I)^{\smile}, \epsilon\Big)^{\downarrow} \;;\; syq\Big((\epsilon \setminus I^{\smile})^{\smile}, \epsilon\Big)^{\downarrow} \\
&= syq\Big((I/\epsilon^{\smile}); syq(\epsilon, I^{\smile}/\epsilon^{\smile})^{\downarrow}, \epsilon\Big)^{\downarrow} && {}^{*}\text{Lemma. 2.4.5 [1 and 2]} \\
&= sqy\Big(I/syq(I^{\smile}/\epsilon^{\smile}, \epsilon)^{\downarrow}; \epsilon^{\downarrow}, \epsilon\Big)^{\downarrow} \\
&= syq\Big(I/(I^{\smile}/\epsilon^{\smile})^{\smile}, \epsilon\Big)^{\downarrow} \\
&= syq\Big(I/(\epsilon \setminus I), \epsilon\Big)^{\downarrow}.
\end{aligned}$$

Based on the relations represented in Definition. 4.1.1, the visual representation of the above definitions can be illustrated as the following diagram: As previously stated, G and

M represent the sets of *objects* and *attributes*, respectively. Also, $\mathcal{L}^G$ is denoted the *relational power set* of *objects* and $\mathcal{L}^M$ is denoted the *relational power set* of *attributes*.
This diagram is later expanded with other relational concepts.

[1]Aforementioned, we have also the relation ϵ (epsilon) between a *set* (for example A) and its *powerset* $\mathcal{P}$(A) (set of all subsets of A), i.e., $\epsilon_{(a,M)}$ = True if a $\in$ M or False otherwise, or simply $\epsilon_{(a,M)}$ = (a in M). In the other words, ϵ: G $\longrightarrow$ 2^G ($M \longrightarrow 2^M$).

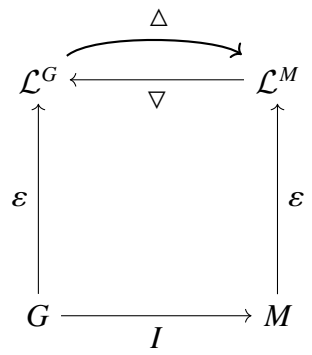


Figure 4.1: Visual representation of relational concepts - 1

4.1.1 Partial Identity of Formal Concepts

As mentioned earlier, not all pairs of objects and attributes are included in the concepts and are not used in creating the lattice.
Using up-triangle and down-triangle, we can check which sets are concepts by generating the *partial identities*. Finding concepts can be done by focusing only on the object part ($\mathbb{I} \sqcap \triangle;\triangledown$) or the attribute part ($\mathbb{I} \sqcap \triangledown;\triangle$). These partial identities based on objects and attributes are also known as *extents* and *intents*, respectively.
So, in this section our focus is on the definitions of partial identities of formal concepts both on boolean and multi-valued contexts.

Previous work on relation algebraic proceeds toward generating partial identities using regular relation of identity, up-triangle, and down-triangle. In fuzzy area, a partial identity $i_G(I)$ is characterized on the *relational power* of G (objects). Correspondingly, $i_M(I)$ is characterized on the *relational power* of M (attributes) [1, 10]. Now, we can summarize the partial identities as below [1]:

Definition 4.1.3. We define two relations $i_G : L_G \to L_G$, and $i_M : L_M \to L_M$ by

$$i_G = \mathbb{I}_{L^G} \sqcap \triangle; \triangledown,$$

Since i_G is a crisp relation, equally we have:

$$i_G(A, A) = 1 \Leftrightarrow A^{\triangle\triangledown} = \mathrm{A}.$$

$i_G(A, A)$ only represents the set of *objects* that are concepts.

$$i_M = \mathbb{I}_{L^M} \sqcap \triangledown; \triangle,$$

Also since i_M is a crisp relation, equally we have:

$$i_M(B, B) = 1 \Leftrightarrow B^{\nabla\triangle} = \mathrm{B}.$$

$i_M(B, B)$ only represents the set of *attributes* that are concepts.

So, using *extents* or *intents* (or *partial identities*), we can say that which sets are concepts.
All partial identities are addressed in the implementation part of this research.

Definition 4.1.4. Before starting the definition of i_{GM}, it is worthy to mention some additional constructions in Arrow categories that we used throughout the thesis. Considering the relational product of two objects A and B (A × B) and two crisp relations π: A × B → A and ρ: A × B → B, we have:

1. $\pi^\smile;\pi \sqsubseteq \mathbb{I}_A$,
2. $\rho^\smile;\rho \sqsubseteq \mathbb{I}_B$,
3. $\mathbb{I}_{L^G \times L^M} = \pi;\pi^\smile \sqcap \rho;\rho^\smile$.

Definition 4.1.5. Operation $\circledleft$ is called *pairing* or *fork*. The reason behind this naming is simple. As an example, considering two relations M and N, (M $\circledleft$ N) relates an element a ∈ A with a pair (b, c) ∈ B × C iff (a, b) ∈ M and (a, c) ∈ N. So, this construction pairs two elements that are related to *a* by M resp. N.
So, we have the definition of $\circledleft$ as follows:

$$M \circledleft N = M;\pi^\smile \sqcap N;\rho^\smile.$$

It is worth noting that $swap = \rho \circledleft \pi$.

Also, we have the definitions of $\circledright$ and $\otimes$ as below:

- $M \circledright N = \pi; M \sqcap \rho; N$,
- $M \otimes N = \pi; M;\pi^\smile \sqcap \rho; N;\rho^\smile$.

Lemma 4.1.2. In addition we have more properties as:

- $(M \circledleft N)^\smile = M^\smile \circledright N^\smile$,
- $(M \circledright N)^\smile = M^\smile \circledleft N^\smile$,
- $M \otimes N = \pi; M \circledleft \rho; N$,

- $M \otimes N = M; \pi^{\smile} \circledcirc N; \rho^{\smile}$
- $(M \otimes N)^{\smile} = M^{\smile} \otimes N^{\smile}$,
- $\mathbb{I}_{A\times B} = \mathbb{I}_A \otimes \mathbb{I}_B$.

Proofs of properties in Lemma. 4.1.2 can be found in [53].

Definition 4.1.6. Considering I: G $\longrightarrow$ M be a *relation*, ϵ: G $\rightarrow L^G$ (M $\rightarrow L^M$), $i_{GM} : L^G \times L^M \longrightarrow L^G \times L^M$, and π and ρ are referred to *projection relations* as π: $\mathcal{P}(G) \times \mathcal{P}(M) \rightarrow \mathcal{P}(G)$ and ρ: $\mathcal{P}(G) \times \mathcal{P}(M) \rightarrow \mathcal{P}(M)$.
The relation i_{GM} refers to those pairs of sets (A, B) which are concepts:

$$i_{GM} = \mathbb{I}_{L^G\times L^M} \sqcap (\vartriangle \otimes \triangledown); \text{swap. [1]}$$

Therefore, using Definition. 4.1.4 i_{GM} equals to:

$$i_{GM} = \mathbb{I} \sqcap \pi; \vartriangle; \rho^{\smile} \sqcap \rho; \triangledown; \pi^{\smile}.$$

In the case of concrete relations and since i_{GM} is a crisp relation, we have:

$$i_{GM}\Big((A, B), (A, B)\Big) = 1 \Leftrightarrow A^{\vartriangle} = \text{B} \wedge B^{\triangledown} = \text{A}.$$

With consideration of new definitions, the expansion of the diagram in Fig. 4.1 is illustrated in the diagram of Fig. 4.2.

It is also noteworthy that by applying three new equations of i_G, i_M, and i_{GM}, the derived concepts are the same as applying the classic equations defined in the Section. 3.1.3 of Chapter. 3 (which I mentioned again in Definition. 4.1.3 and 4.1.6 as crisp relations). It means that all derived pairs of objects and attributes in concepts are identically the same, and then they can be used to draw the corresponding lattice. Remarkably, this further proves the fact that we can equally use a *relational algebraic approach* to define the *Formal concept analysis* (FCA).

So far, we have indicated the assertion of $(A_1, B_1) \sqsubseteq (A_2, B_2)$ iff $A_1 \sqsubseteq A_2$ or $B_2 \sqsubseteq B_1$ for two concepts of (A_1, B_1) and (A_2, B_2) in a lattice in Chapter. 3. Consequently, we can choose one of the $A_1 \sqsubseteq A_2$ or $B_2 \sqsubseteq B_1$ assertions because of their equivalent derived result. Next lemma includes a proof of the fact that $A_1 \sqsubseteq A_2$ equals to $B_2 \sqsubseteq B_1$ in the *relational* approach.

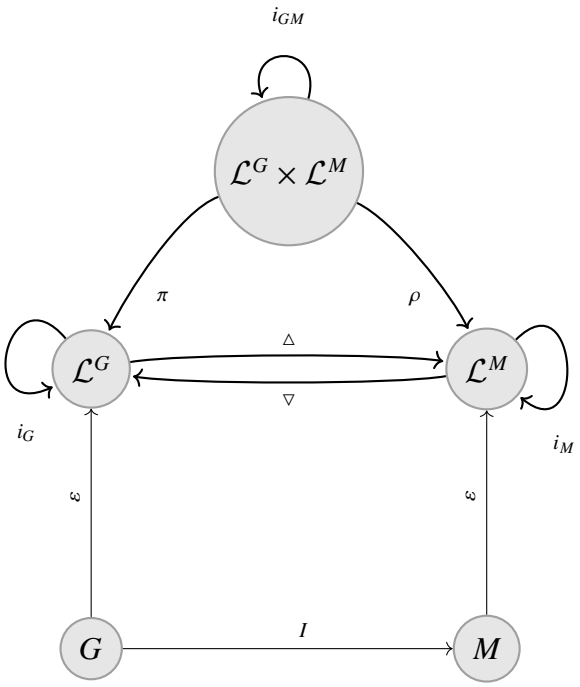


Figure 4.2: Visual representation of relational concepts - 2

Lemma 4.1.3. Considering definitions of 4.1.6 and 4.1.6, for A $\sqsubseteq$ G and B $\sqsubseteq$ M, we have the algebraic definition of $A^{\triangle}$ = B iff $B^{\triangledown}$ = A as below:

$$i_{GM} \,;\, \pi \,;\, \mathrm{e} \,;\, \pi^{\smile} \,;\, i_{GM} = i_{GM} \,;\, \rho \,;\, e^{\smile} \,;\, \rho^{\smile} \,;\, i_{GM} \qquad (1)$$

Proof 4.1.2. Now, we have the proof of the equation presented in the Lemma. 4.1.3.
Let R: G $\rightarrow$ M be a relation.
At the beginning, considering the following equations we show that $\triangle^{\smile}$; e ; $\triangle \sqsubseteq e^{\smile}$:

$$\begin{aligned} e; \triangle; \epsilon^{\smile} &= (\epsilon \setminus \epsilon); syq(R^{\smile}/\epsilon^{\smile}, \epsilon)^{\downarrow}; \epsilon^{\smile} \\ &= (\epsilon \setminus \epsilon); (R^{\smile}/\epsilon^{\smile})^{\smile} \\ &= (\epsilon \setminus \epsilon)(\epsilon \setminus R) \\ &\sqsubseteq \epsilon \setminus R. \end{aligned}$$

And we have:

$$\begin{aligned}\epsilon \setminus R &= (R^{\smile}/\epsilon^{\smile})^{\smile} \\ &= syq(R^{\smile}/\epsilon^{\smile}, \epsilon)^{\downarrow}; \epsilon^{\smile} \\ &= \triangle; \epsilon^{\smile}.\end{aligned}$$

So, we have e ; $\triangle$; $\epsilon^{\smile} \sqsubseteq \epsilon \setminus$ R and $\epsilon \setminus$ R = $\triangle$; $\epsilon^{\smile}$. Then we can conclude that:

$$e; \triangle; \epsilon^{\smile} \sqsubseteq \triangle; \epsilon^{\smile}.$$

Now, by adding two $\triangle^{\smile}$ to the both sides ($\triangle$ is *univalent*):

$$\triangle^{\smile}; e; \triangle; \epsilon^{\smile} \sqsubseteq \triangle^{\smile}; \triangle; \epsilon^{\smile}.$$

Then, we have:

$$\triangle^{\smile}; e; \triangle; \epsilon^{\smile} \sqsubseteq \epsilon^{\smile}$$

So:

$$\begin{aligned}\triangle^{\smile}; e; \triangle &\sqsubseteq \epsilon^{\smile} \setminus \epsilon^{\smile} \\ &\sqsubseteq (\epsilon \setminus \epsilon)^{\smile} \\ &\sqsubseteq e^{\smile}.\end{aligned}$$

Finally, we get:

$$\triangle^{\smile}; e; \triangle \sqsubseteq e^{\smile}. \qquad (2)$$

Now, using left side of equation (1):

$$\begin{aligned}i_{GM}; \pi; e; \pi^{\smile}; i_{GM} &= i_{GM}; i_{GM}^{\smile}; \pi; e; \pi^{\smile}; i_{GM}; i_{GM} && \text{add } i_{GM}; i_{GM}^{\smile} \\ &\sqsubseteq i_{GM}; \rho; \triangle^{\smile}; \pi^{\smile}; \pi; e; \pi^{\smile}; \pi; \triangle; \rho^{\smile}; i_{GM} && \text{definition of } i_{GM} \\ &\sqsubseteq i_{GM}; \rho; \triangle^{\smile}; e; \triangle; \rho^{\smile}; i_{GM} \\ &\sqsubseteq i_{GM}; \rho; e^{\smile}; \rho^{\smile}; i_{GM} && (2)\end{aligned}$$

In the same way, the proof of the opposite side of the equation (1) can be concluded.

4.1.2 Splitting of Partial Identity

Please note that all three partial identities i_G, i_M and i_{GM} are *crisp* since they are defined by using just crisp relations. As a consequence we can form their *splitting*, i.e., we use the relation c_{GM}: $C_{GM} \to L^G \times L^M$ (resp. c_G: $C_G \to L^G$ and c_M: $C_M \to L_M$) characterized by $i_{GM} = c_{GM}^{\smile}$; c_{GM} and c_{GM} ; $c_{GM}^{\smile} = \mathbb{I}_{C_{GM}}$ (resp. $i_G = c_G^{\smile}$; c_G and c_G ; $c_G^{\smile} = \mathbb{I}_{C_G}$, also, $i_M = c_M^{\smile}$; c_M and c_M ; $c_M^{\smile} = \mathbb{I}_{C_M}$) [1]. So, we can see that c_M splits i_M, c_G splits i_G, and c_{GM} splits i_{GM}.

Bijective maps between relations C_{GM}, C_G, C_M, i.e., between the sets of formal concepts given as a pair, by their object set, and their attribute set, are defined as below [1].

Definition 4.1.7. We define three relations b_G^{GM}: $C_G \to C_{GM}$, b_M^{GM}: $C_M \to C_{GM}$, and b_G^M: $C_G \to C_M$ by:

1. $b_G^{GM} = c_G; (\mathbb{I}_{L^G} \circledcirc \triangle); c_{GM}^{\smile}$,

2. $b_M^{GM} = c_M; (\triangledown \circledcirc \mathbb{I}_{L^M}); c_{GM}^{\smile}$,

3. $b_G^M = b_G^{GM}; (b_M^{GM})^{\smile}$.

It is worthy of note that $(\triangledown \circledcirc \mathbb{I}_{L^M}) = (\triangledown; \pi^{\smile} \sqcap \rho^{\smile})$ and $(\mathbb{I}_{L^G} \circledcirc \triangle^{\smile}) = \pi \sqcap \rho; \triangle^{\smile}$.

Theorem 4.1.1. The relations b_G^{GM}, b_M^{GM}, and b_G^M are *bijective* maps because each element in either c_M or c_G is paired exactly with an element in c_{GM} and vice versa.

PROOF. First of all, we start with the equation $b_M^{GM}; (b_M^{GM})^{\smile} = \mathbb{I}_B$ which we want to show that it is *total* and *injective*. Using the definition of b_G^M, we can derive the third assertion immediately once we have indicated the first and second assertion.

So, we have:

$$
\begin{aligned}
\mathbb{I}_B &= b_M^{GM}; (b_M^{GM})^\smile \\
&= c_M; (\nabla \circledS \mathbb{I}_{L^M}); c_{GM}^\smile; (c_M; (\nabla \circledS \mathbb{I}_{L^M}); c_{GM}^\smile)^\smile \\
&= c_M; (\nabla \circledS \mathbb{I}_{L^M}); c_{GM}^\smile; c_M^\smile; (\nabla \circledS \mathbb{I}_{L^M})^\smile; c_{GM} \\
&= c_M; (\nabla \circledS \mathbb{I}_{L^M}); c_{GM}^\smile; c_{GM}; (\nabla \circledS \mathbb{I}_{L^M})^\smile; c_M^\smile \\
&= c_M; (\nabla \circledS \mathbb{I}_{L^M}); i_{GM}; (\nabla \circledS \mathbb{I}_{L^M})^\smile; c_M^\smile && \text{Definition. 4.1.6 and 4.1.7} \\
&= c_M; (\nabla; \pi^\smile \sqcap \rho^\smile); (\mathbb{I} \sqcap \pi; \triangle; \rho^\smile \sqcap \rho; \nabla; \pi^\smile); (\pi; \nabla^\smile \sqcap \rho); c_M^\smile \\
&= c_M; (\nabla; \pi^\smile \sqcap \rho^\smile); (\pi; \triangle; \rho^\smile \sqcap \rho; \nabla; \pi^\smile \sqcap \mathbb{I}); (\pi; \nabla^\smile \sqcap \rho); c_M^\smile \\
&= c_M; ((\nabla; \pi^\smile \sqcap \rho^\smile); \pi; \triangle; \rho^\smile \sqcap (\nabla; \pi^\smile \sqcap \rho^\smile); \rho; \nabla; \pi^\smile \sqcap \nabla; \pi^\smile \sqcap \rho^\smile); (\pi; \nabla^\smile \sqcap \rho); c_M^\smile \\
&= c_M; (\nabla; \triangle; \rho^\smile \sqcap \nabla; \pi^\smile \sqcap \nabla; \pi^\smile \sqcap \rho^\smile); (\pi; \nabla^\smile \sqcap \rho); c_M^\smile && \text{remove the same assertions} \\
&= c_M; (\nabla; \triangle; \rho^\smile \sqcap \nabla; \pi^\smile \sqcap \rho^\smile); (\pi; \nabla^\smile \sqcap \rho); c_M^\smile && \text{distribution} \\
&= c_M; (\nabla; \triangle; \rho^\smile; (\pi; \nabla^\smile \sqcap \rho) \sqcap \nabla; \pi^\smile; (\pi; \nabla^\smile \sqcap \rho) \sqcap \rho^\smile; (\pi; \nabla^\smile \sqcap \rho)); c_M^\smile \\
&= c_M; (\nabla; \triangle \sqcap \nabla; \nabla^\smile \sqcap \mathbb{I}); c_M^\smile \\
&= c_M; (\nabla; \triangle \sqcap \mathbb{I}); c_M^\smile \\
&= c_M; c_M^\smile; c_M; c_M^\smile \\
&= \mathbb{I}_B.
\end{aligned}
$$

So, the second assertion in Definition. 4.1.7 is *total* and *injective*.

In order to show that the relation is also *univalent*, we have to show the following equation:

$$(c_G; (\mathbb{I}_{L^G} \circledS \triangle); c_{GM}^\smile)^\smile; c_G \; ; (\mathbb{I}_{L^G} \circledS \triangle); c_{GM}^\smile = \mathbb{I}_{L^G \times L^M}$$

Or equivalently:

$$(b_G^{GM})^\smile; b_G^{GM} = \mathbb{I}_{L^G \times L^M}.$$

Starting from left-hand side of the equation, we have:

$$
\begin{aligned}
&(c_M; (\mathbb{I}_{L^M} \circledcirc \triangle); c_{GM}^{\smile})^{\smile}; c_M; (\mathbb{I}_{L^M} \circledcirc \triangle); c_{GM}^{\smile} \\
&= c_M^{\smile}; (\mathbb{I}_{L^M} \circledcirc \triangle)^{\smile}; c_{GM}; c_M; (\mathbb{I}_{L^M} \circledcirc \triangle); c_{GM}^{\smile} \\
&= c_{GM}; (\mathbb{I}_{L^M} \circledcirc \triangle)^{\smile}; c_M^{\smile}; c_M; (\mathbb{I}_{L^M} \circledcirc \triangle); c_{GM}^{\smile} && \text{since } c_G^{\smile}; c_G = i_G \\
&= c_{GM}; (\mathbb{I}_{L^M} \circledcirc \triangle)^{\smile}; i_M; (\mathbb{I}_{L^M} \circledcirc \triangle); c_{GM}^{\smile} && \text{Definition. 4.1.7} \\
&= c_{GM}; (\nabla; \pi^{\smile} \sqcap \rho^{\smile})^{\smile}; i_M; (\nabla; \pi^{\smile} \sqcap \rho^{\smile}); c_{GM}^{\smile} && \text{since } i_M = \mathbb{I}_{L^M} \sqcap \nabla ; \triangle \\
&= c_{GM}; (\nabla^{\smile}; \pi \sqcap \rho); (\nabla; \triangle \sqcap \mathbb{I}); (\nabla; \pi^{\smile} \sqcap \rho^{\smile}); c_{GM}^{\smile} \\
&\sqsubseteq c_{GM}; (\pi; \nabla^{\smile}; \nabla; \triangle \sqcap \pi; \nabla^{\smile} \sqcap \rho); (\nabla; \pi^{\smile} \sqcap \rho^{\smile}); c_{GM}^{\smile} \\
&\sqsubseteq c_{GM}; (\pi; \nabla^{\smile}; \nabla; \triangle \sqcap \pi; \nabla^{\smile} \sqcap \rho); (\nabla; \pi^{\smile}) \sqcap (\pi; \nabla^{\smile}; \nabla; \triangle \sqcap \pi; \nabla^{\smile} \sqcap \rho); (\rho^{\smile}); c_{GM}^{\smile} \\
&\sqsubseteq c_{GM}; (\pi; \nabla^{\smile}; \nabla; \triangle; \nabla; \pi^{\smile} \sqcap \pi; \nabla^{\smile}; \nabla; \pi^{\smile} \sqcap \rho; \nabla; \pi^{\smile} \sqcap \pi; \nabla^{\smile}; \nabla; \triangle; \rho^{\smile} \sqcap \pi; \nabla^{\smile}; \rho^{\smile} \sqcap \rho; \rho^{\smile}); c_{GM}^{\smile} \\
&\sqsubseteq c_{GM}; (\pi; \nabla^{\smile}; \nabla; \pi^{\smile} \sqcap \rho; \nabla; \pi^{\smile} \sqcap \pi; \nabla^{\smile}; \nabla; \triangle; \rho^{\smile} \sqcap \rho; \rho^{\smile}); c_{GM}^{\smile} \\
&\sqsubseteq c_{GM}; (\pi; \pi^{\smile} \sqcap \rho; \nabla; \pi^{\smile} \sqcap \pi; \triangle; \rho^{\smile} \sqcap \rho; \rho^{\smile}); c_{GM}^{\smile} \\
&= c_{GM}; (\pi; \triangle; \rho^{\smile} \sqcap \rho; \nabla; \pi^{\smile} \sqcap \mathbb{I}); c_{GM}^{\smile} && \text{Definition. 4.1.6} \\
&= c_{GM}; i_{GM}; c_{GM}^{\smile} && \text{since } i_{GM} = c_{GM}; c_{GM}^{\smile} \\
&= c_{GM}; c_{GM}^{\smile}; c_{GM}; c_{GM}^{\smile} \\
&= \mathbb{I}
\end{aligned}
$$

In fact, in the second part of the proof we show that $c_{GM}; c_{GM}^{\smile}; c_{GM}; c_{GM}^{\smile} = \mathbb{I}_{L^G \times L^M}$. Now, the opposite inclusion $\left(\mathbb{I}_{L^G \times L^M} = c_{GM}; c_{GM}^{\smile}; c_{GM}; c_{GM}^{\smile} \right)$ is formed as below, which shows that this relation is also *surjective*:

$$
\begin{aligned}
&\mathbb{I}_{L^G \times L^M} = c_{GM}; c_{GM}^{\smile}; c_{GM}; c_{GM}^{\smile} && \text{since } c_{GM}^{\smile}; c_{GM} = i_{GM} \\
&= c_{GM}; i_{GM}; c_{GM}^{\smile} && \text{Definition. 4.1.6} \\
&= c_{GM}; (\mathbb{I} \sqcap \pi; \triangle; \rho^{\smile} \sqcap \rho; \nabla; \pi^{\smile}); c_{GM}^{\smile} \\
&= c_{GM}; (\pi; \pi^{\smile} \sqcap \rho; \rho^{\smile} \sqcap \pi; \triangle; \rho^{\smile} \sqcap \rho; \nabla; \pi^{\smile}); c_{GM}^{\smile} \\
&= c_{GM}; (\rho; (\nabla; \pi^{\smile} \sqcap \rho^{\smile}) \sqcap \pi; \pi^{\smile} \sqcap \pi; \triangle; \rho^{\smile}); c_{GM}^{\smile}
\end{aligned}
$$

$\sqsubseteq c_{GM}; ((\pi; \pi^\smile \sqcap \pi; \triangle; \rho^\smile); (\pi; \triangledown^\smile \sqcap \rho) \sqcap \rho); (\triangledown; \pi^\smile \sqcap \rho^\smile); c_{GM}^\smile$ since $(\triangledown \circledless \mathbb{I}_{L^M}) = \triangledown; \pi^\smile \sqcap \rho^\smile$

$= c_{GM}; (\pi; \triangle \sqcap \pi; \triangledown^\smile \sqcap \rho); (\triangledown \circledless \mathbb{I}_{L^M}); c_{GM}^\smile$

$\sqsubseteq c_{GM}; (\pi; \triangledown^\smile \sqcap \rho); ((\triangledown; \pi^\smile; \rho^\smile); \pi; \triangle \sqcap \mathbb{I}_{L^M}); (\triangledown \circledless \mathbb{I}_{L^M}); c_{GM}^\smile$ since $(\triangledown \circledless \mathbb{I}_{L^M}) = \triangledown; \pi^\smile \sqcap \rho^\smile$

$= c_{GM}; (\triangledown \circledless \mathbb{I}_{L^M})^\smile; ((\triangledown; \pi^\smile; \rho^\smile); \pi; \triangle \sqcap \mathbb{I}_{L^M}); (\triangledown \circledless \mathbb{I}_{L^M}); c_{GM}^\smile$

$= c_{GM}; (\triangledown \circledless \mathbb{I}_{L^M})^\smile; (\triangledown; \triangle; \mathbb{I}_{L^M}); (\triangledown \circledless \mathbb{I}_{L^M}); c_{GM}^\smile$ since $(\triangledown; \triangle; \mathbb{I}_{L^M}) = i_M$

$= c_{GM}; (\triangledown \circledless \mathbb{I}_{L^M})^\smile; i_M; (\triangledown \circledless \mathbb{I}_{L^M}); c_{GM}^\smile$

$= c_{GM}; i_M; c_{GM}^\smile$ since $i_M = c_M^\smile; c_M$

$= c_{GM}; c_M^\smile; c_M; c_{GM}^\smile$

The second assertion follows analogously.

Taking into consideration the ordering between concepts to generate the lattice, in the next lemma we define some order relations. Now, based on these relations, we can distinguish between pairs that are concepts or those that form only attribute or object pairs. The definition of the order relations can be found as below [1].

Definition 4.1.8. Considering $\Omega = (\epsilon \setminus \epsilon)^\downarrow$, we define three relations E_{GM}: $C_{GM} \to C_{GM}$, E_G: $C_G \to C_G$, and E_M: $C_M \to C_M$ on $L^G \times L^M$, L^G, and L^G respectively by

1. $E_{GM} = c_{GM}\ ;(\pi\ ;\Omega\ ;\pi^\smile \sqcap \rho\ ;\Omega^\smile\ ;\rho^\smile); c_{GM}^\smile$,
2. $E_G = c_G\ ;\Omega\ ;c_G^\smile$,
3. $E_M = c_M\ ;\Omega\ ;c_M^\smile$.

It is easy to verify that the three relations defined above are indeed orderings.

Visualization of the defined objects and relations is illustrated in the Fig. 4.3. As a summary of the graph in a few sentences, i_{GM} shows the pairs G and M giving all pairs on $L^G \times L^M$. The partial identities i_G and i_M only contain objects and attributes of all pairs, respectively. Regarding split functions, c_{GM} shows the only pairs that are concepts. Similarly, c_G and c_M only contain objects and attributes of all those pairs that are concepts, respectively. So, we can see that c_{GM}, c_G, and c_M only split those pairs that are concepts.
About order relations, E_{GM} is the strict order relation of those pairs that are concepts, including objects and attributes. As a matter of fact, they are in relation iff the object sets are

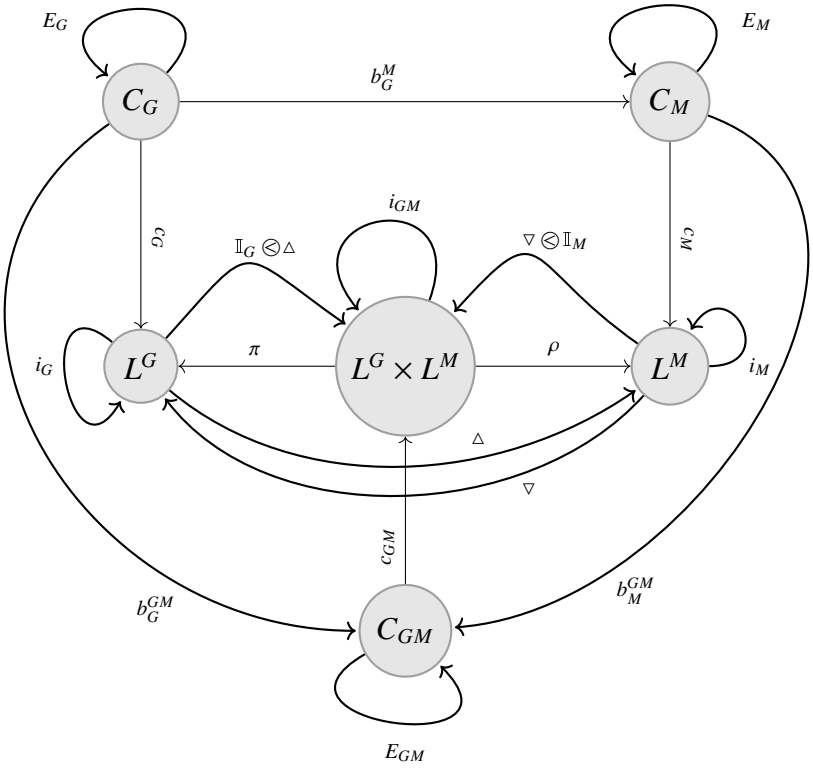


Figure 4.3: Extensive visualization of the new relations

included π ; Ω ; $\pi^\smile$ and the attribute sets are in the opposite inclusion ρ ; $\Omega^\smile$; $\rho^\smile$. E_G is the strict order relation of the objects of all the pairs that are concepts and similarly, E_M is the strict order relation of the attributes of all the pairs that are concepts.

As we mentioned before, split functions including c_{GM}, c_G, and c_M take only those pairs that are concepts. The same applies to order relations (Ω) which only gives us the order on those pairs that are concepts. In other words, all concepts formed by the context are in order using a hierarchical relation which is called *concept lattice*. Now, we define the lattice order on concepts based on the object and attribute sets in the Definition. 4.1.8(1). Lemma. 4.1.4 describes this ordering which is determined by the order on one of its components only i.e., object part (c_{GM} ; π ; Ω ; $\pi^\smile$; $c_{GM}^\smile$) or attribute part (c_{GM} ; ρ ; $\Omega^\smile$; $\rho^\smile$; $c_{GM}^\smile$) for the order relation E_{GM} generates the same result.

Lemma 4.1.4.

$$E_{GM} = c_{GM}; \pi; \Omega; \pi^\smile; c_{GM}^\smile = c_{GM}; \rho; \Omega^\smile; \rho^\smile; c_{GM}^\smile$$

Proof 4.1.3. For the proof of Lemma. 4.1.4, we already had:

$$\begin{aligned}
E_{GM} &= c_{GM}; (\pi; \Omega; \pi^\smile \sqcap \rho; \Omega^\smile; \rho^\smile); c_{GM}^\smile && \text{distribution} \\
&= (c_{GM}; \pi; \Omega; \pi^\smile \sqcap c_{GM}; \rho; \Omega^\smile; \rho^\smile); c_{GM}^\smile && \text{distribution} \\
&= c_{GM}; \pi; \Omega; \pi^\smile; c_{GM}^\smile \sqcap c_{GM}; \rho; \Omega^\smile; \rho^\smile; c_{GM}^\smile && \text{add } c_{GM}^\smile \text{ ; } c_{GM} \\
&= c_{GM}; c_{GM}^\smile; c_{GM}; c_{GM}^\smile; c_{GM}; \pi; \Omega; \pi^\smile; c_{GM}^\smile \sqcap c_{GM}; \rho; \Omega^\smile; \rho^\smile; c_{GM}^\smile \\
&= c_{GM}; c_{GM}^\smile; c_{GM}; \pi; \Omega; \pi^\smile; c_{GM}^\smile; c_{GM}; c_{GM}^\smile \sqcap c_{GM}; \rho; \Omega^\smile; \rho^\smile; c_{GM}^\smile \\
&= c_{GM}; c_{GM}^\smile; c_{GM}; \rho; \Omega^\smile; \rho^\smile; c_{GM}^\smile; c_{GM}; c_{GM}^\smile \sqcap c_{GM}; \rho; \Omega^\smile; \rho^\smile; c_{GM}^\smile \\
&= c_{GM}; \rho; \Omega^\smile; \rho^\smile; c_{GM}^\smile.
\end{aligned}$$

Now, starting from left assertion, we proved the right assertion. Also, we used $\mathbb{I}_{C_{GM}} = c_{GM}; c_{GM}^\smile$ and also $\pi; \Omega; \pi^\smile = \rho; \Omega^\smile; \rho^\smile$ equations in the middle of the proof.
Correspondingly, the other side of the equation follows analogously.

In the following lemma, we show that the maps represented in the Definition. 4.1.7 are all monotonic.

Lemma 4.1.5. Let E_F and E_M are two *order* relations. The bijective relation $c_M \,;\, (\triangledown \circledcirc \mathbb{I}_M) \,;\, c_{GM}^{\smile}$ (or $C_M \to C_{GM}$ in Fig. 4.3) is called *monotone* iff:

$$c_M \,;\, (\triangledown \circledcirc \mathbb{I}_M) \,;\, c_{GM}^{\smile} \,;\, E_{GM} \sqsubseteq E_M \,;\, c_M \,;\, (\triangledown \circledcirc \mathbb{I}_M); c_{GM}^{\smile}$$

or equivalently:

$$E_M \sqsubseteq c_M \,;\, (\triangledown \circledcirc \mathbb{I}_M) \,;\, c_{GM}^{\smile} \,;\, E_{GM} \,;\, \Big(c_M; (\triangledown \circledcirc \mathbb{I}_M); c_{GM}^{\smile}\Big)^{\smile}.$$

Proof 4.1.4. Now, we have the proof of the Lemma. 4.1.5.
We already mentioned that:

$$\begin{aligned}
&c_M; (\triangledown \circledcirc \mathbb{I}_M); c_{GM}^{\smile}; E_{GM}; \Big(c_M; (\triangledown \circledcirc \mathbb{I}_M); c_{GM}^{\smile}\Big)^{\smile} \\
&= c_M; (\triangledown \circledcirc \mathbb{I}_M); c_{GM}^{\smile}; E_{GM}; c_M^{\smile}; (\triangledown \circledcirc \mathbb{I}_M)^{\smile}; c_{GM} \\
&= c_M; (\triangledown \circledcirc \mathbb{I}_M); c_{GM}^{\smile}; E_{GM}; c_{GM}; (\triangledown \circledcirc \mathbb{I}_M)^{\smile}; c_M^{\smile} && \text{Lemma. 4.1.4} \\
&= c_M; (\triangledown \circledcirc \mathbb{I}_M); c_{GM}^{\smile}; c_{GM}; \rho; \Omega^{\smile}; \rho^{\smile}; c_{GM}^{\smile}; c_{GM}; (\triangledown \circledcirc \mathbb{I}_M)^{\smile}; c_M^{\smile} \\
&\sqsubseteq c_M; (\triangledown \circledcirc \mathbb{I}_M); \rho; \Omega^{\smile}; \rho^{\smile}; (\triangledown \circledcirc \mathbb{I}_M)^{\smile}; c_M^{\smile} \\
&\sqsubseteq c_M; (\triangledown \circledcirc \mathbb{I}_M); \rho; \rho^{\smile}; \Omega^{\smile}; (\triangledown \circledcirc \mathbb{I}_M)^{\smile}; c_M^{\smile} \\
&= c_M; \Big((\triangledown \circledcirc \mathbb{I}_M); \rho\Big); \Omega^{\smile}; \Big((\triangledown \circledcirc \mathbb{I}_M); \rho\Big)^{\smile}; c_M^{\smile} \\
&= c_M; \Omega^{\smile}; c_M^{\smile} && \text{Definition. 4.1.8} \\
&= E_M.
\end{aligned}$$

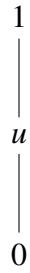
We can have the same proof to show that the map $c_G; (\mathbb{I}_G \circledcirc \triangle); c_{GM}^{\smile}$ is *monotonic*.

Now, we demonstrate our work using an example derived from the implementation part of this research. Before we generate attribute implications, we have to form pairs of concepts using provided *i* and *c* versions of *partial identities* and *splittings* of them, respectively.

Example 4.1.1. A lattice of the truth values containing three elements 1, u, and 0 is considered in this example. As shown in the below diagrams, we have operation $\twoheadrightarrow$ which is a $*$ operation along with its adjoint.
The generated context (relation *I* which is *I: Persons* $\to$ *Sports*) is created using two sets *Persons* = {Duro, Fero, Jano, Miso, Palo} and *Sports* = {Hockey, Football, Volleyball, Basketball, Frisbee}:

$*$	0	u	1
0	0	0	0
u	0	0	1
1	0	1	2

$\twoheadrightarrow$	0	u	1
0	2	2	2
u	1	2	2
1	0	1	2



$$\boldsymbol{I} = \begin{array}{c} \\ \textit{Duro} \\ \textit{Fero} \\ \textit{Jano} \\ \textit{Miso} \\ \textit{Palo} \end{array} \begin{array}{c} \begin{array}{ccccc} \textit{Hockey} & \textit{Football} & \textit{Volleyball} & \textit{Basketball} & \textit{Frisbee} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & u & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Also, above relation I is shown in our implementation part (first tab) using Java programming language given as 2-dimensional arrays of $\mathcal{L}$-values similar to the above matrix. It is worthy to note that only the first letter referring to a person or sport can be used here as an abbreviation. For example V/u indicates that "Volleyball" is in a *certain fuzzy set* with degree u. We used the whole name of sports (last tab) in our implementation for a better understanding.

For example in our implementation we have {Hockey $\mapsto$ 0, Football $\mapsto$ 1, Volleyball $\mapsto$ u, Basketball $\mapsto$ 0, Frisbee $\mapsto$ 0} denotes the fuzzy set of sports that contains "Volleyball" with the degree u, "Football" with the full degree 1, and "Hockey" with the degree 0 (i.e., not at all), etc.

Some obtained concepts are indicated as below:

$\Big($ {Hockey $\mapsto$ 0, Football $\mapsto$ 0, Volleyball $\mapsto$ 0, Basketball $\mapsto$ 0, Frisbee $\mapsto$ 0},
{ Hockey $\mapsto$ 0, Football $\mapsto$ 0, Volleyball $\mapsto$ 0, Basketball $\mapsto$ u, Frisbee $\mapsto$ 0 } $\Big)$

$$\Big(\{\text{Hockey} \mapsto 0, \text{Football} \mapsto 0, \text{Volleyball} \mapsto 0, \text{Basketball} \mapsto \text{u}, \text{Frisbee} \mapsto 1\},$$
$$\{\ \text{Hockey} \mapsto 0, \text{Football} \mapsto 0, \text{Volleyball} \mapsto 0, \text{Basketball} \mapsto \text{u}, \text{Frisbee} \mapsto \text{u}\ \} \Big)$$

$$\Big(\{\text{Hockey} \mapsto 0, \text{Football} \mapsto \text{u}, \text{Volleyball} \mapsto \text{u}, \text{Basketball} \mapsto 1, \text{Frisbee} \mapsto 1\},$$
$$\{\ \text{Hockey} \mapsto \text{u}, \text{Football} \mapsto 1, \text{Volleyball} \mapsto \text{u}, \text{Basketball} \mapsto 1, \text{Frisbee} \mapsto 1\ \} \Big)$$

4.2 Attribute Implications

Let us now focus on only attributes on a Boolean context as well as a multi-valued context. In this section, our purpose is to generate some algebraic formulas to construct a set of all attribute implications. Following this, a relational algebraic formula is proposed by the implication basis. Using the mentioned relational approach, a valid implication set is generated that satisfies certain rules of all implications.
An attribute implication is an expression P $\rightarrow$ Q that relates two sets P and Q of attributes and expresses that every object possessing each attribute from P also has each attribute from Q. When (G, M, I) is a formal context and P, Q are subsets of the set M of attributes (i.e., P, Q $\sqsubseteq$ M), then the implication P $\rightarrow$ Q is true (or valid).
Also, we can define attribute implication formally and relation-algebraically as below [1]:

Definition 4.2.1. Formally attribute implication is defined as: P $\rightarrow$ Q holds in a context I: G $\rightarrow$ M if P $\sqsubseteq$ A implies Q $\sqsubseteq$ A for all intents A.

Definition 4.2.2. Relation-algebraically, we can define a relation A_I: $\mathcal{L}^M \rightarrow \mathcal{L}^M$ so that we have (P, Q) $\in A_I$ iff P $\rightarrow$ Q holds in a context I: G $\rightarrow$ M.

More precisely, we can define attribute implication (A_I) algebraically using 5 various alternative formulations using the definition of Ω in Section. 2.4.4.2 on the relation A_I [1]. The purpose of the attribute implication formulas is generating all valid implications from the context.

Theorem 4.2.1. Let consider $\Omega = (\epsilon \setminus \epsilon)^{\downarrow}$ [2], then:

1. $A_I = (i_M; \Omega^{\smile}) \setminus \Omega^{\smile}$,
2. $A_I = \triangledown; \triangle; \Omega^{\smile}$,
3. $A_I = \big((I \not* \epsilon^{\smile}) \setminus (I \not* \epsilon^{\smile})\big)^{\downarrow}$,
4. $A_I = \triangledown; \Omega; \triangledown^{\smile}$,
5. $A_I = (c_M; \Omega^{\smile}) \setminus (c_M; \Omega^{\smile})$.

It is worthy to note that the operations of *right residuals* in the formulation of the number *3* are all *star-based*. It is the only formulation where the star-based operations are used directly. In all other formulations, the star-based operations are used in the definition of the *triangle* operations.
All these formulations are implemented using *Java* programming and for all these 5 versions of formulas, we got the same results or relations in implementation.

Now, we show the proof of the second and third formulations, as well as the fifth one. You can find other proofs in [1].

Proof 4.2.3 (2) To prove formulation of number 2, we assume that formulation of number 1 is approved and can be used. Then we have, $A_I = \triangledown; \triangle; \Omega^{\smile} = (i_M; \Omega^{\smile}) \setminus \Omega^{\smile}$. So that we first show $i_M; \Omega^{\smile}; \triangledown; \triangle; \Omega^{\smile}; \epsilon^{\smile} \sqsubseteq \epsilon^{\smile}$:

$$\begin{aligned} i_M; \Omega^{\smile}; \triangledown; \triangle; \Omega^{\smile}; \epsilon^{\smile} &\sqsubseteq i_M; \Omega^{\smile}; \triangledown; \triangle; \epsilon^{\smile}, \\ &\sqsubseteq i_M; \triangledown; \triangle; \Omega^{\smile}; \epsilon^{\smile}, \\ &\sqsubseteq i_M; \triangledown; \triangle; \epsilon^{\smile} = i_M^{\smile}; \triangledown; \triangle; \epsilon^{\smile}, \\ &\sqsubseteq (\triangledown; \triangle)^{\smile}; \triangledown; \triangle; \epsilon^{\smile} = i_M^{\smile}; \epsilon^{\smile}, \\ &\sqsubseteq \epsilon^{\smile}. \end{aligned}$$

So, we have:

$$i_M; \Omega^{\smile}; \triangledown; \triangle; \Omega^{\smile}; \epsilon^{\smile} \sqsubseteq \epsilon^{\smile}$$

[2] Ω is the order on the power set object.

Then we can conclude:

$$i_M; \Omega^\smile; \triangledown; \triangle; \Omega^\smile \sqsubseteq \epsilon^\smile / \epsilon^\smile$$

Since we have:

$$i_M; \Omega^\smile; \triangledown; \triangle; \Omega^\smile = (i_M; \Omega^\smile; \triangledown; \triangle; \Omega^\smile)^\downarrow$$

And, also:

$$(\epsilon^\smile / \epsilon^\smile)^\downarrow = \Omega^\smile$$

Briefly, we can show this by:

$$i_M; \Omega^\smile; \triangledown; \triangle; \Omega^\smile = (i_M; \Omega^\smile; \triangledown; \triangle; \Omega^\smile)^\downarrow \sqsubseteq (\epsilon^\smile / \epsilon^\smile)^\downarrow = \Omega^\smile$$

Since the right side of the above equation is crisp, then we can conclude:

$$i_M; \Omega^\smile; \triangledown; \triangle; \Omega^\smile \sqsubseteq \Omega^\smile$$

And, therefore:

$$\triangledown; \triangle; \Omega^\smile \sqsubseteq (i_M; \Omega^\smile) \setminus \Omega^\smile.$$

Also, consider the following computation:

$$\begin{aligned}
(\triangledown; \triangle)^\smile; \big((i_M; \Omega^\smile) \setminus \Omega^\smile\big) &= (\triangledown; \triangle)^\smile; (\triangledown; \triangle); (\triangledown; \triangle)^\smile; \big((i_M; \Omega^\smile) \setminus \Omega^\smile\big) \\
&= i_M; (\triangledown; \triangle)^\smile; \big((i_M; \Omega^\smile) \setminus \Omega^\smile\big) && \triangledown\ ; \triangle \text{ is a map} \\
&\sqsubseteq i_M; \Omega^\downarrow; \big((i_M; \Omega^\smile) \setminus \Omega^\smile\big) \\
&\sqsubseteq \Omega^\smile
\end{aligned}$$

So that we have: $(\triangledown; \triangle)^\smile; \big((i_M; \Omega^\smile) \setminus \Omega^\smile\big) \sqsubseteq \Omega^\smile$. Then, we have:

$$((i_M; \Omega^{\smile}) \setminus \Omega^{\smile}) \sqsubseteq \triangledown; \triangle; \Omega^{\smile}.$$

Proof 4.2.3 (3) Using the formulation number 2, we can prove the formulation number 3. So, starting from left side, we can conclude the right side in equation $((I/\epsilon^{\smile}) \setminus (I/\epsilon^{\smile}))^{\downarrow} = \triangledown; \triangle; \Omega^{\smile}$. This proof is straightforward. The whole detail can be found in [1].

Proof 4.2.3 (5) To prove the formulation number 5, considering $K \sqsubseteq A_I$ we have:

$$\begin{aligned}
K \sqsubseteq A_I &\Leftrightarrow K \sqsubseteq (i_M; \Omega^{\smile}) \setminus \Omega^{\smile} && \text{by formulation (1)} \\
&\Leftrightarrow i_M; \Omega^{\smile}; K \sqsubseteq \Omega^{\smile} && \text{since } i_M = c_M^{\smile}\,;\, c_M \\
&\Leftrightarrow c_M^{\smile}; c_M; \Omega^{\smile}; K \sqsubseteq \Omega^{\smile} \\
&\Leftrightarrow c_M; \Omega^{\smile}; K \sqsubseteq c_M; \Omega^{\smile} \\
&\Leftrightarrow K \sqsubseteq (c_M; \Omega^{\smile}) \setminus (c_M; \Omega^{\smile})
\end{aligned}$$

And, finally we can derive:

$$A_I = (c_M; \Omega^{\smile}) \setminus (c_M; \Omega^{\smile}).$$

The relation A_I in the formulation number 5 is an embedding between all attribute subsets from the context. For a better understanding, we refer to the implementation part of this research (operation emd in Relation). This operation deletes all 0 rows from intents or extents so that it will be easier to read them.

All the five represented versions of the formulations are used in our implementation for *classical* (*2-valued*) and *3-valued* examples to generate attribute implications. The details are described in Chapter. 5. As mentioned earlier, the final relations generated by all five versions of the formulas are the same.

Example 4.2.1. Since the number of the attribute implications is too big (the actual number is 31,566 attribute implications) to list them all, this example just focus on one of them generated by Example. 4.1.1 [1].

It is worth mentioning that we used the same example in the implementation section of this research.

$$\{H \mapsto 0, F \mapsto 0, V \mapsto 0, B \mapsto 1, Fr \mapsto u\} \longmapsto \{H \mapsto 0, F \mapsto 0, V \mapsto u, B \mapsto 1, Fr \mapsto 0\}$$

As a simplified representation for easier understanding, a more comprehensible form of implication is shown below:

Hockey	*Football*	*Volleyball*	*Basketball*	*Frisbee*
0	0	0	1	u
↓	↓	↓	↓	↓
0	0	u	1	0

This implication indicates that if it is unknown that a person likes *Frisbee*, but definitely likes *Basketball*, then it is also unknown whether the person likes *Volleyball*.

As already mentioned in the Example. 4.2.1, the set of all valid attribute implications of a given context is usually huge. Therefore, we are usually interested in a basis of implications, i.e., a small set of implications from which we can generate the whole set of implications naturally. In order to do so, we use so-called *Duquenne-Guigues* or *canonical* basis [1, 34]. As a prerequisite to define *Duquenne-Guigues* basis, we need to define *pseudo-closed* concept.

Definition 4.2.3. In *Duquenne-Guigues* basis, the elements are generated by *pseudo-closed* sets of attributes [1], i.e., using a set A so that $A \neq A^{\nabla\triangle}$ and $B^{\nabla\triangle} \sqsubseteq A$ for all pseudo-closed $B \subsetneq A$.

We simply assume that the set of *pseudo-closed* sets exists . The *relational* modelling of *pseudo-closed set* is presented in the below definition. More details can be found in [1].

Definition 4.2.4. Consider a crisp relation p: $L^M \to L^M$. This relation is called the set of *pseudo-closed* sets if:

$$p = \mathbb{I}_{L^M} \sqcap ((p; \Omega_s) \setminus (\nabla; \triangle; \Omega)) \sqcap (\nabla; \triangle)_s \; ; \; \pi_{L^M L^M}.$$

In this definition, if M = N, p(M, N) = 1 can be the only case. This is because of the *identity* relation on the right-hand side of the definition of p. Therefore, p(M) = 1 is written instead of p(M, N) = 1 and $M = N$.

Therefore, *Duquenne-Guigues* basis can be provided by { P $\rightarrow P^{\nabla\triangle}$ | P is pseudo-closed }, i.e., by the relation p ; ∇ ; $\triangle$ [1].

Definition 4.2.5. Our example has 227 pseudo-closed sets [1]. One of these pseudo-closed sets is the set { H $\mapsto$ 0, F $\mapsto$ 0, V $\mapsto$ 0, B $\mapsto$ u, C $\mapsto$ u}. This leads to the implication

{ H $\mapsto$ 0, F $\mapsto$ 0, V $\mapsto$ 0, B $\mapsto$ u, C $\mapsto$ u } $\mapsto$ { H $\mapsto$ u, F $\mapsto$u, V $\mapsto$ u, B $\mapsto$ u, C $\mapsto$ u }.

as one of the 227 elements of the *Duquenne-Guigues* basis that is a part of all 31,566 attribute implications of our example.

Chapter 5

Implementation

This chapter is discussed the implementation phase of the algebraic framework for $\mathcal{L}$-fuzzy relations. Among all programming languages, Java is more suited with our purpose as a high-level, class-based, object-oriented programming language. Starting with why we choose Java for our implementation, we go into more detail about our classes to gain a deeper understanding of the implementation phase of this research. Meanwhile, all the methods are fully explained along with their applications and possible outputs.

5.1 Java Language Selection

Java is a general purpose programming language that was developed and released in 1995. It is influenced by C++. The main motivation behind Java was to make a programming language that allows its users to "write once, run everywhere". The difference between the way Java worked, and other programming languages at that time was revolutionary. Instead of being compiled into a specific machine code, the Java compiler turns code into something called Bytecode, which is then interpreted by a software called the Java Runtime Environment (JRE), or the Java Virtual Machine (JVM). The JRE acts as a virtual computer that interprets Bytecode and translates it for the host computer. Because of this, Java code can be written the same way for many platforms [40].

Along with other important functional features of Java programming like *type safety* and *functional capabilities*, being an *Object Oriented Programming* (*OOP*) is a very significant feature for us in our implementation. *OOP* languages model the world by *classes*. Every class has its own attributes, also known as fields, and a number of methods that might change the state of those fields, also known as behavior. An object is a real instance of a certain class. Similarly to the real world, classes can be part of other classes that are usually less restricted. Therefore, they can inherit their certain fields and methods from their

parent class i.e., superclass. Actually every class defined in Java's API or the programmer defines themselves inherits directly or indirectly from the foundation class Object, that is the common superclass of every other class written in Java [40]. In our implementation we focus on classifying our work by generating some classes such as *Relation*, *RelationDisplay* or *FuzzyAlgebra* classes. This is one of the most significant reasons for choosing the Java language to implement our work.

So far we have discussed mostly the features of Java programming language. Now, we will go through a brief summary of Java strengths again. Java strengths rely on an OOP language that is supported by a huge class library API that has almost everything the developer needs. In addition to that, Java is secure and it prevents potential malicious code to have access to certain platform features and APIs which could be exploited by malware, such as accessing the local filesystem, running arbitrary commands, or accessing communication networks. Java has been there for around 25 years and that makes it adopted by a big community of developers [40].

5.2 Implementation Details

5.2.1 System Overall Description

This implementation is generated to prove and indicate the accuracy and correctness of the mentioned five algebraic formulas in Section. 4.2 of the Chapter. 4 to $\mathcal{L}$-fuzzy formal concept analysis. It consists of several different classes and an interface to properly display the results to users. To put it in more detail, classes of Main, Relation, and FuzzyAlgebra are the core of this implementation containing all main creator methods for the *GUI* (Graphical User Interface). It is also worth mentioning that two classes *RelationDisplay* and *OperationDisplay* are generated to assist the representations of *relations* and *FuzzyAlgebra* (operations), respectively.

We dive deeper into some major classes and their methods of our implementation in detail in the following sections.

5.2.2 Classes

5.2.2.1 Class of "*Relation*"

The class "*Relation*" is the core of the implementation in our work. It consists of all primary methods such as *meet*, *join*, *composition*, *transpose*, *residuals*, *symmetricQuotient* which are explained in the second chapter of this thesis. Moreover, All methods associated with the fuzzy idea are included in this class.
It is worthwhile to note that each of the methods in fuzzy idea work exactly the same as *meet*, *impl*, *composition*, *leftResidual*, *rightResidual*, *symmetricQuotient* by just replacing *meet* with *star*, *impl* with *starImpl*, *composition* with *starComposition*, *leftResidual* with *starLeftResidual*, *rightResidual* with *starRightResidual*, and *symmetricQuotient* with *starSymmtricQuotient* to correspond to the fuzzy idea.

An overview of some main methods in the class "*Relation*" are provided as below:

1. ***identity (algebra, dom)***:
   - **Type**: Relation.
   - **Method Description**: returns a square relation with 1 on the main diagonal and 0 elsewhere.
   - **Return Value**: *identity* relation with specified dimensions.
2. ***scalar (algebra, dom, n)***:
   - **Type**: Relation.
   - **Method Description**: is the same as identity() with one difference; the diagonal elements can be any int value (*n*) and other elements should be 0 too.
   - **Return Value**: *scalar* relation with specified dimensions.
3. ***top (algebra, dom, cod)***:
   - **Type**: Relation.
   - **Method Description**: returns a matrix with greatest value of elements (1).
   - **Return Value**: *top* relation with specified dimensions.
4. ***bot (algebra, dom, cod)***:
   - **Type**: Relation.

- **Method Description**: returns a matrix with smallest value of elements (0).
- **Return Value**: *bottom* relation with specified dimensions.

5. ***ideal (algebra, dom, cod, n)***:

   - **Type**: Relation.
   - **Method Description**: returns a matrix with the ideal constant value (n) for all elements. In other words, all elements can be constant n.
   - **Return Value**: *ideal* relation with specified dimensions.

6. ***pi (algebra, a, b)***:

   - **Type**: Relation.
   - **Method Description**: this method creates a matrix ($a*b$) as rows and a as columns ($a*b \rightarrow a$) using the a and b parameters. The returned matrix consists only elements of 0 and 1, so that it contains 1 as element for those indexes which represents a as their columns and 0 elsewhere.
   - **Return Value**: *pi* relation consists values of 0 and 1. The dimensions of the returned relation depend on the input parameters of a and b using *ProductSetObject* class.

7. ***rho (algebra, a, b)***:

   - **Type**: Relation.
   - **Method Description**: this method creates a matrix ($a*b$) as rows and b as columns ($a*b \rightarrow b$) using the a and b parameters. The returned matrix consists only elements of 0 and 1, so that it contains 1 as element for those indexes which represents b as their columns and 0 elsewhere.
   - **Return Value**: *rho* relation consists values of 0 and 1. The dimensions of the returned relation depend on the input parameters of a and b using *ProductSetObject* class.

8. ***iota (algebra, a, b)***:

   - **Type**: Relation.
   - **Method Description**: this method creates a matrix a as rows and $a+b$ as columns ($a \rightarrow a+b$) using the a and b parameters. The returned matrix consists only elements of 0 and 1, so that it contains 1 as element for those indexes where row and column represents same element from a and 0 elsewhere.

- **Return Value**: *iota* relation consists values of 0 and 1. The dimensions of the returned relation depend on the input parameters of *a* and b using *SumSetObject* class.

9. ***kappa (algebra, a, b)***:

   - **Type**: Relation.
   - **Method Description**: this method creates a matrix *b* as rows and *a+b* as columns ($b \rightarrow a+b$) using the *a* and *b* parameters. The returned matrix consists only elements of 0 and 1, so that it contains 1 as element for those indexes where row and column represents same element from *b* and 0 elsewhere.
   - **Return Value**: *kappa* relation consists values of 0 and 1. The dimensions of the returned relation depend on the input parameters of *a* and *b* using *SumSetObject* class.

10. ***epsilon (algebra, dom)***:

    - **Type**: Relation.
    - **Method Description**: this method establishes a relation between *dom* and the *power set* of *dom* (all subsets derived from *dom*), such that if a particular *dom* is in a particular subset of dom, the epsilon relation contains 1 and 0 otherwise. Therefore, the dimension of the epsilon relation is size of dom to the $2^{Size\ of\ dom}$.
    - **Return Value**: *epsilon* relation consists values of 0 and 1. The dimensions of the returned relation depend on the input parameter of *dom* using *PowerSetObject* class.

11. ***composition (relation)***:

    - **Type**: Relation.
    - **Method Description**: the *relative multiplication* of two relations is calculated by this method in a way that the (x,z) element would be true in the final relation if and only if there exists one element y such that (x, y) is true in the first relation and (y, z) is true in the other one.
    - **Return Value**: *composition* relation of two relations.

12. ***transpose ( )***:

    - **Type**: Relation.

- **Method Description**: switches rows and columns (*dom* and *cod* or *objects* and *attributes*) of the relation and generates a new relation.
- **Return Value**: *transposed* relation of a relation.

13. ***leftResidual (relation)***:

- **Type**: Relation.
- **Method Description**: considering two relations R and S, this method calculates R \ S which is the *left residual*. In other words, it shows the *greatest* of all relations X such that (X.R) is the subset of S.
- **Return Value**: *leftResidual* relation of two relations.

14. ***rightResidual (relation)***:

- **Type**: Relation.
- **Method Description**: considering two relations R and S, this method calculates R / S which is the *right residual*. In other words, it shows the *greatest* of all relations X such that (R.X) is the subset of S.
- **Return Value**: *rightResidual* relation of two relations.

15. ***symmetricQuotient (relation)***:

- **Type**: Relation.
- **Method Description**: generates the *intersection* of the *left* and *right* residual relations to two relations R and S. Considering R: X $\rightarrow$ Y and S: X $\rightarrow$ Z. Then this method relates an element y from Y to an element z from Z when y and z have the same set of inverse image.
- **Return Value**: *symmetricQuotient* relation of two relations.

Also, an overview of some main methods in the class of "*FuzzyAlgebra*" are provided as below:

1. ***up (int m)***:

- **Type**: int.
- **Method Description**: if the element does not equal to 0 (or in other words, bigger than 0), this method returns 1, otherwise it returns 0.

- **Return Value**: an integer consists values of only 0 or 1, and finally the created relation only contains the values 0 and 1.

2. ***down (int m)***:

   - **Type**: int.
   - **Method Description**: if the element equals to 1, this method returns 1, otherwise it returns 0.
   - **Return Value**: an integer consists values of only 0 or 1, and finally the created relation only contains the values 0 and 1.

3. ***intersection (set1, set2)***:

   - **Type**: set.
   - **Method Description**: this method generates the *meet* of the elements belonging to two sets.

4. ***union (set1, set2)***:

   - **Type**: set.
   - **Method Description**: this method generates the *join* of the elements belonging to two sets.

5. ***impl (set1, set2, list)***:

   - **Type**: set.
   - **Method Description**: this method determines the greatest element (x) such that the *meet* of x and element1 from set1 is *less* or *equal* to element2 from set2. X can be any element from *lattice*.

5.3 Interface General Overview

The implementation is also included a GUI as a clearer view for a better understanding of our work. This interface is generated by a *JFrame* to display the main results. The main area of the *JFrame* has multiple tabs which are made using *JTabbedPane* class of *Java* to be able to switch between tabs by clicking on the specific given titles of mentioned tabs.

The interface in the implementation is included three dominant groups of data consists of *Context*, *FuzzyAlgebra*, and *Attribute Implications* which have to be shown in several tabs. To make our work more comprehensible, two sets of *Persons* and *Sports* are considered as the *object* and *attribute* sets (named them *cod* and *dom* in coding), respectively.

5.3.1 ”Context” tab

The first tab of the GUI represents the *2-valued* (*regular*) and *3-valued* (*fuzzy*) relations with elements names of { ”0”, ”1” } and { ”0”, ”U”, ”1” }, respectively. So, the first tab generates the *context* of *persons*, *sports*, and their likings for both the *regular* and *fuzzy* ideas. This work is done by two sets of input data naming *table* and *table-F* to help creation of regular and fuzzy contexts, respectively. As I mentioned earlier, data entries in table-F contains ambiguity. This is because any entry that is not 0 or 1 is considered ambiguous data.

table = { {0, 1, 0, 1, 0}, {1, 0, 1, 0, 1}, {1, 0, 0, 0, 1}, {0, 1, 0, 1, 0}, {0, 0, 1, 0, 0} },

table-F = { {0, 2, 0, 2, 0}, {2, 0, 1, 0, 2}, {2, 0, 0, 0, 2}, {0, 2, 0, 2, 0}, {0, 0, 2, 0, 0} }.

Using *RelationDisplay* class, *Context* tab displays two relations of I and I-F (F refers to the Fuzzy idea) are formed in the *Main* class of implementation and shown in 2-valued (Figure. 5.1) and 3-valued (Figure. 5.2) tabs, respectively.

Also, we have two sets of **truthHA** = { ”0”, ”1” } and **chain3HA** = { ”0”, ”u”, ”1” } as truth degrees for Heyting algebra of regular (2-valued) and fuzzy (3-valued) ideas, respectively. This means that all values in the tables in the Context tab come from these two sets, depending on whether regular or fuzzy ideas are represented.

5.3.2 ”Fuzzy Algebra” tab

This tab shows the *Heyting Algebra* or the same *Fuzzy Algebra* (including Fuzzy idea). As shown in Figure. 5.3, five operations of *meet*, *join*, *impl*, *star*, and *starImpl* are used to indicate the results in separate tables for the second tab of the interface. The elements of the mentioned table on which operations are applied come from *FuzzyAlgebra* class. We implemented this part by *OperationDisplay* class which is very similar to *RelationDisplay* class.
It is worth mentioning that *star* and *starImpl* operations are only applied on *Fuzzy* idea.

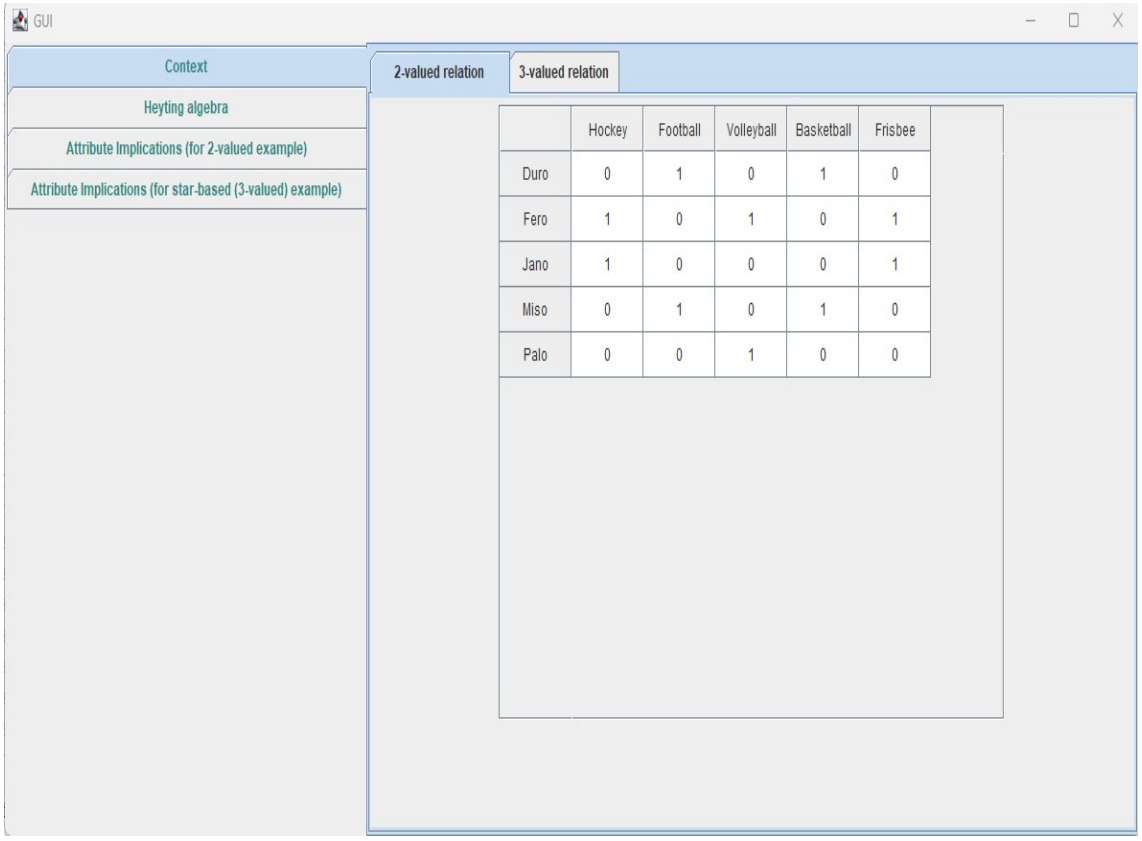


Figure 5.1: Context tab representing 2-valued (I) relation - GUI

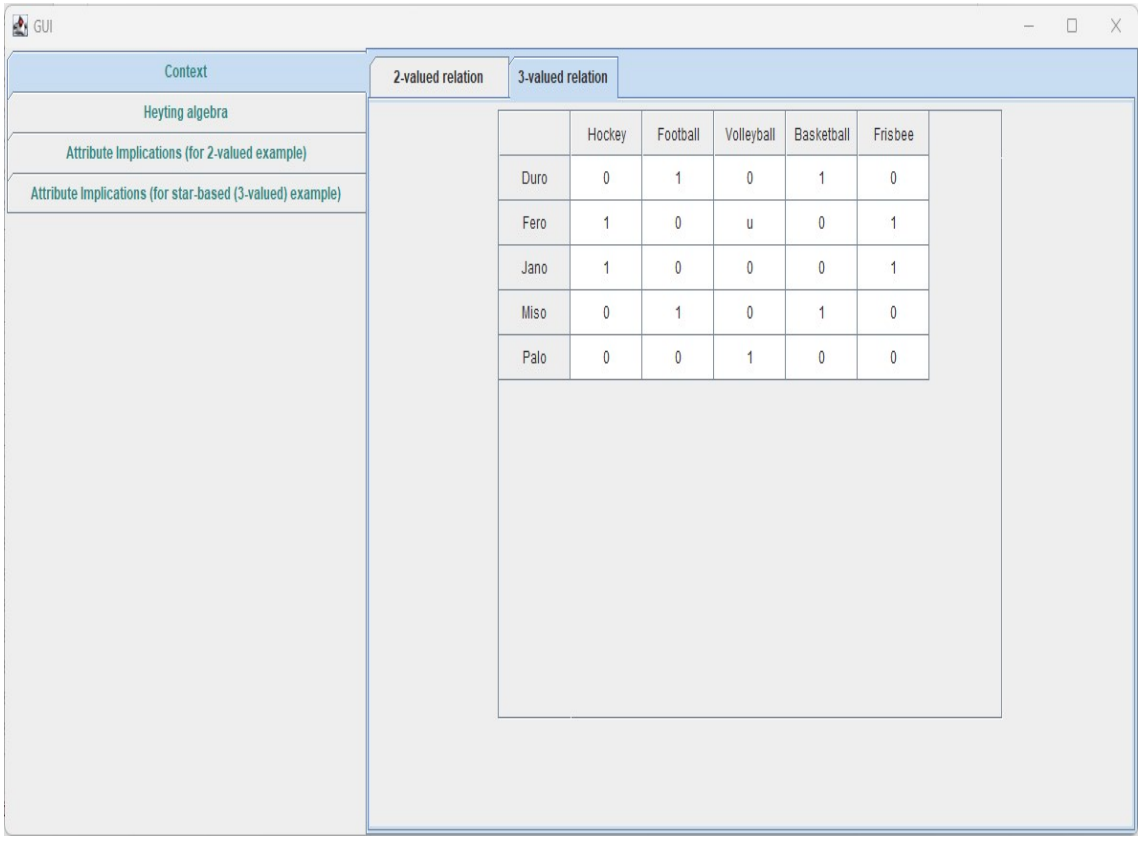


Figure 5.2: Context tab representing 3-valued (I-F) relation - GUI

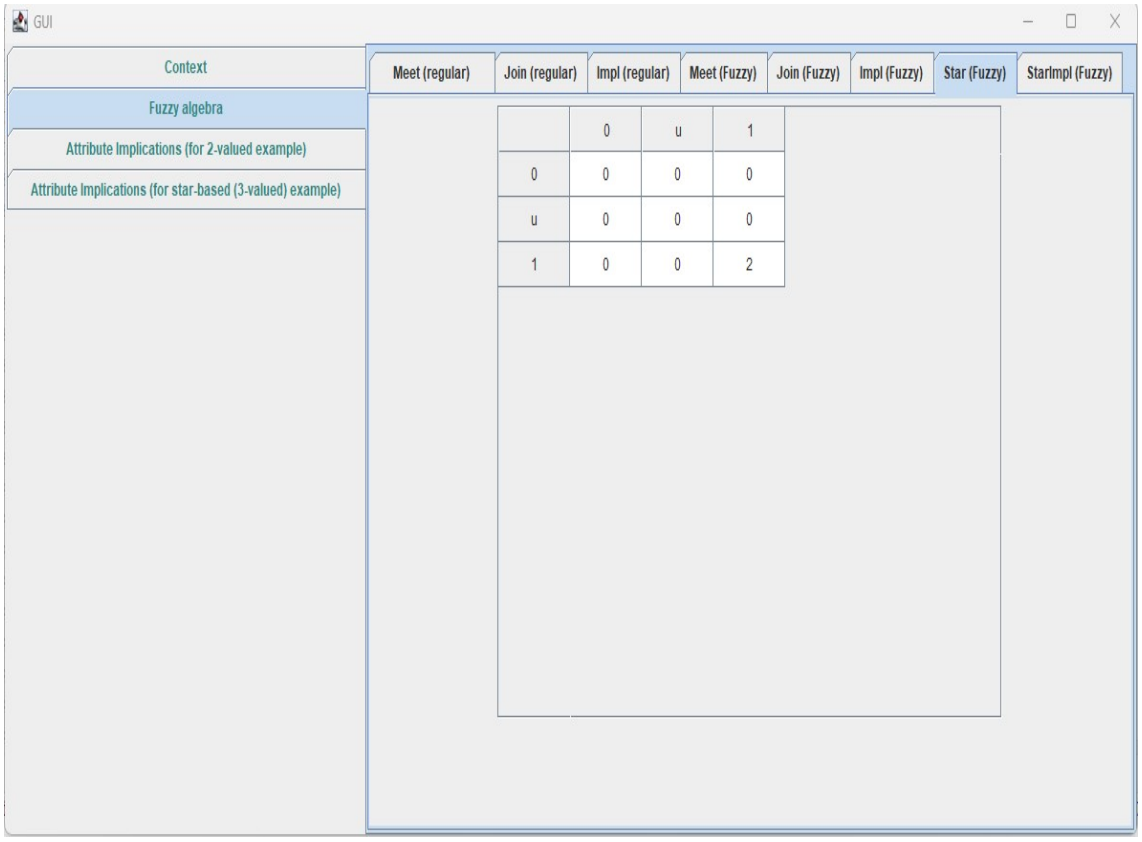


Figure 5.3: FuzzyAlgebra tab representing 5 different operations in both regular and fuzzy ideas on FuzzyAlgebra elements - GUI

5.3.3 ”Attribute Implications (2-valued example)” tab

Using 5 various versions of the attribute implication formulas presented in Definition. 4.2.1, this tab shows all *attribute implications* only for 2-valued or regular example. As the 5 tables display, we have exactly the same results (tables) by using the mentioned formulations.

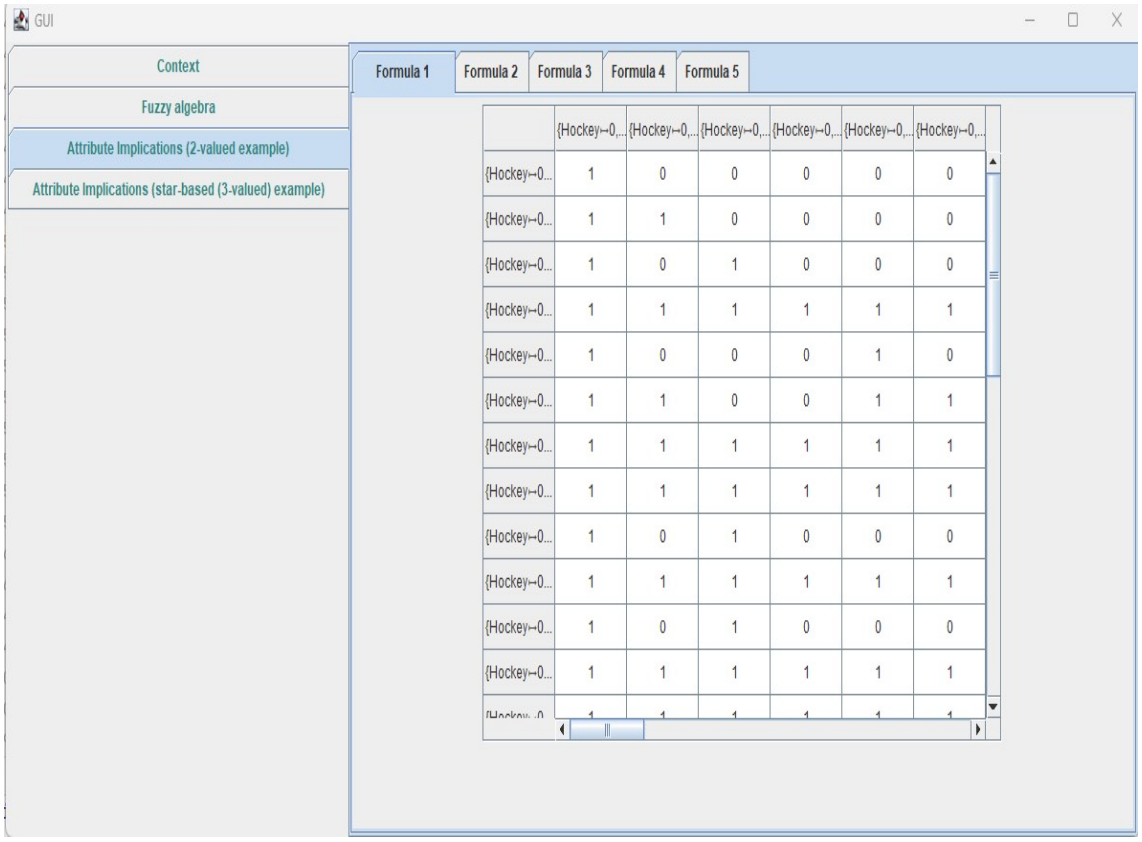


Figure 5.4: Attribute Implications (2-valued example) tab - GUI

5.3.4 ”Attribute Implications (3-valued example)” tab

Similar to the tab expressed in the Section. 5.3.3, this tab displays the attribute implications for 3-valued (fuzzy) example created by 5 versions of attribute implication formulas. Also, same as attribute implications for 2-valued example, we have exactly the same tables for 3-valued example.

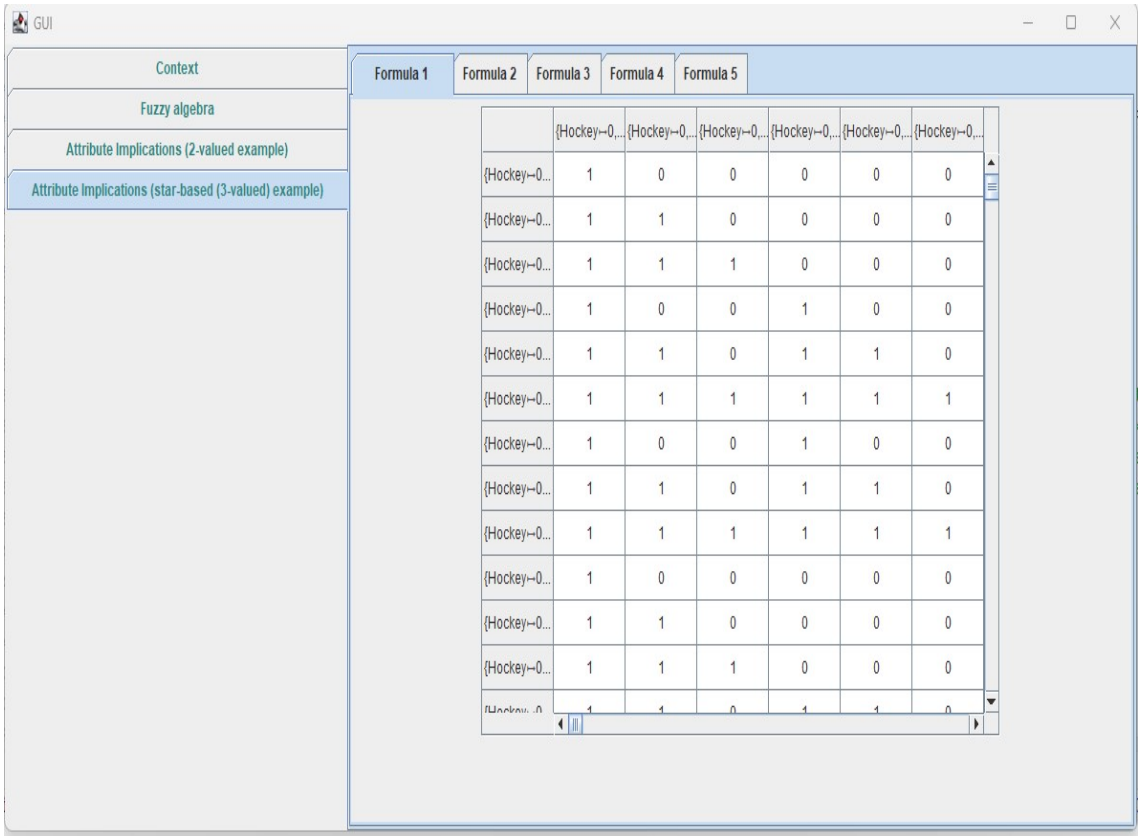


Figure 5.5: Attribute Implications (3-valued example) tab - GUI

As shown in Figures. 5.4 and 5.5, we do not get rid of the uncertainty available in the input data. We just try to handle it so that we can extract patterns of the attribute implications from the ambiguous data.

5.4 UML Diagram

At the end of this thesis, a UML diagram including whole classes, methods, fields, properties, and constructors is illustrated in the Figure. 5.6. As it is clear in the UML diagram, the classes of *Relation* and *FuzzyAlgebra* have the most significant roles in the implementation containing several fundamental methods. Then using the mentioned fundamental methods, the contexts, attribute implications, etc., are generated in the class of *Main*. Two classes of *RelationDisplay* and *OperationDisplay* are used for representation purposes in the interface as already explained in the Section. 5.2.1.

5.5 Using the Framework

Considering the input data of different users, the final output of attribute implications can be different. In the first step, the input data including persons (objects), sports (attribute), and also table-F sets must be defined according to users' opinions. Even users can choose any other data instead of the values represented in *chain3HA* set as the $\mathcal{L}$ values. Using the mentioned input data, contexts of both regular and fuzzy ideas can be generated. These contexts show the relations between objects and attributes that users already specified.
Next step is creation of Heyting (Fuzzy) algebra. As already explained, in this phase, all operations of *Meet*, *Join*, *Implication*, *Star*, and *starImpl* are made using the provided contexts. We need to have all these operations to generate constitutive relations of attribute implications, i.e., leftResidual or symmetricQuotient (syq).
In the final stage, all $\mathcal{L}$-valued table of attribute implications related to specific inputs are made.

Now, we want to generate patterns or attribute implications according to the users' desired data by this framework. This section can make a clear understanding of the use of this framework with users' own examples. This action starts from *star-based* example in the implementation which belongs to the *Fuzzy* idea of our approach. Therefore, some parts of coding need be replaced by user data starting this section of the main program. First step is to add the lattice $\mathcal{L}$. For example, we would like to use a lattice consisting of 6 elements as explained below.

Draw the lattice: In order to generate the lattice we only need to provide a set of sets that has the same order structure than the irreducible elements (an element is irreducible iff

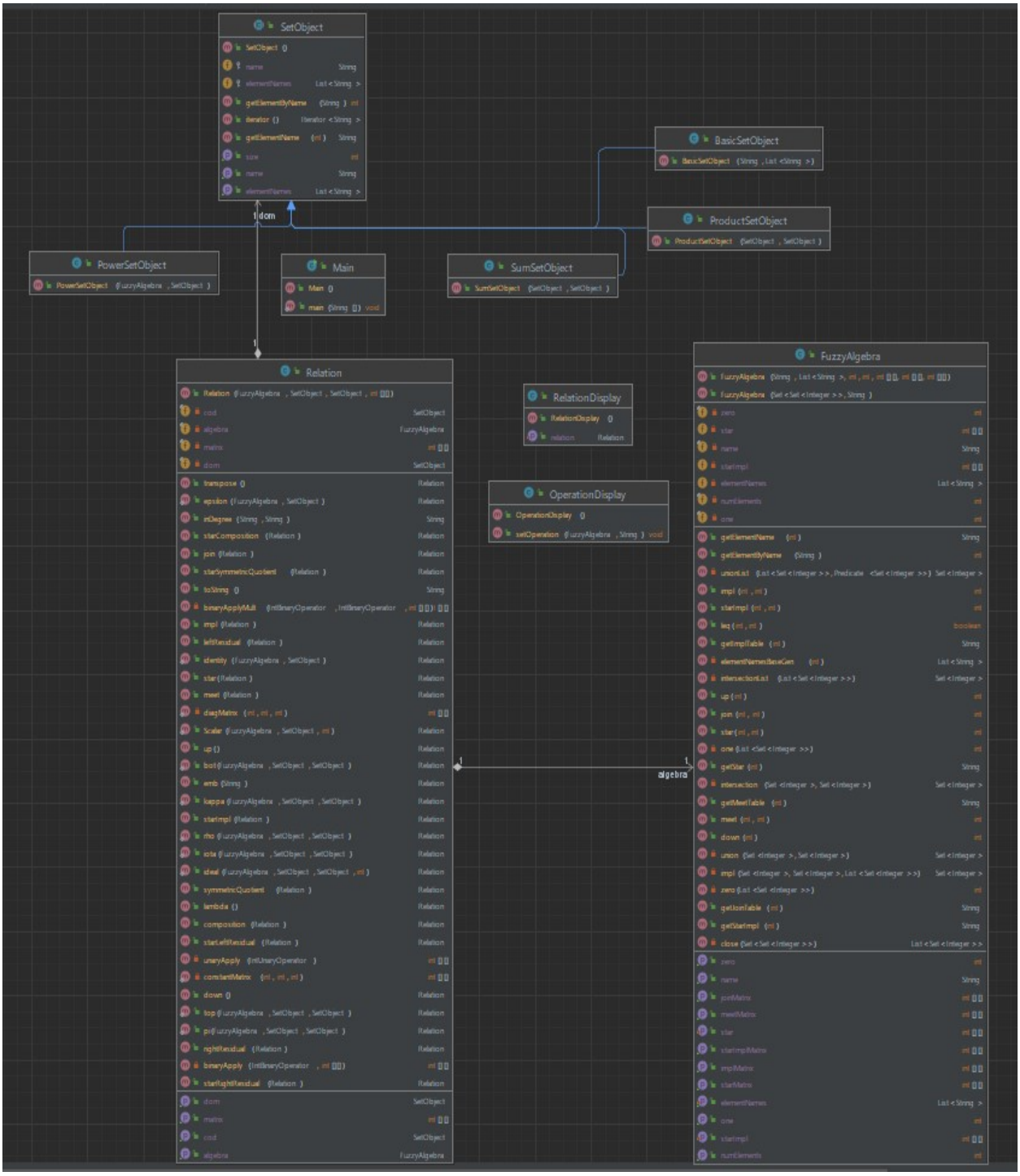


Figure 5.6: UML Diagram

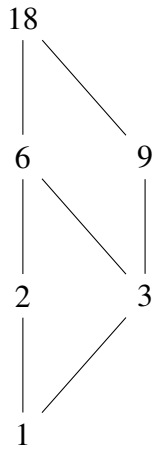


Figure 5.7: Visual representation of the lattice for divisors 18

it is different from 0 and is not the join of two other elements) of the lattice. For example, lattice of divisors of 18 is chosen. This lattice can be generated using the set method by the sets of {1}, {2}, and {2,3}. In this lattice, an element is smaller than another iff the smaller one divides the greater one, evenly. The mentioned lattice contains 6 elements, i.e., 1, 2, 3, 6, 9, and 18. The mentioned lattice is drawn in Figure. 5.7. This can be implemented by:

```
Set<Set<Integer>> generators = new HashSet();
generators.add(Set.of(1));
generators.add(Set.of(2));
generators.add(Set.of(2, 3));
```

in the program. Now we generate the set of those sets in Java and call the corresponding constructor of *FuzzyAlgebra* as below:

```
FuzzyAlgebra chain6HA = new FuzzyAlgebra(generatorsF, "6-valued");
```

It is noteworthy that F refers to Fuzzy idea of our approach in all parts of the coding.
Next we can provide names for the elements of the generated lattice by:

```
chain6HA.setElementNames(Arrays.asList(new String[]
        { "0", "1", "2", "3", "4", "5" }));
```

Define *star* and *starImpl* matrices: In the next step, two matrices of *star* and *starImpl* should be defined. For example, we can define them as below:

```
int[][] star_divisor18 =
{{ 0, 0, 0, 3, 4, 5 }, { 0, 0, 1, 2, 4, 5 }, { 0, 1, 2, 3, 4, 5 }};
```

```
int[][] starImpl_divisor18 =
{{ 2, 2, 2, 4, 4, 4 }, { 1, 2, 2, 4, 5, 5 }, { 0, 1, 2, 3, 3, 5 }};
```

Now we can relate elements of *star_divisor*18 and *starImpl_divisor*18 matrices to the elements of *chain6HA* using *setStar* and *setStarImpl* methods of *FuzzyAlgebra* as shown below:

```
chain6HA.setStar(star_divisor18);
chain6HA.setStarImpl(starImpl_divisor18);
```

Define object and attribute sets: Following step generates the context of object and attribute. All these values can be chosen arbitrarily in this example:

```
BasicSetObject obj_divisor18 = new BasicSetObject("obj_divisor18",
 Arrays.asList(new String[] { "18", "90", "36", "72", "81" }));

BasicSetObject attr_divisor18 = new BasicSetObject("attr_divisor18",
   Arrays.asList(new String[] { "36", "18", "6", "72", "18" }));
```

Define input data: Now is the time to define our desired input data in *table_divisor*18 matrix as follows:

```
int[][] table_divisor18 =
{{ 1, 2, 4, 2, 0 }, { 2, 0, 5, 0, 2 }, { 2, 3, 1, 5, 2 },
{ 4, 2, 1, 2, 3 }, { 0, 5, 2, 3, 0 }};
```

Generate context: Using the class of "*Relation*", a relation among the object, attribute, using *input* data and *chain6HA* can be generated as *I_divisor*18:

```
Relation I_divisor18 =
new Relation(chain6HA, obj_divisor18, attr_divisor18, table_divisor18);
```

Generate some relations based on objects and attributes: *epsilon* and *omega* relations are made in this step using both *objects* (*extents*) and *attributes* (*intents*):

```
Relation epsilon1_divisor18 = Relation.epsilon(chain6HA, obj_divisor18);
Relation epsilon2_divisor18 = Relation.epsilon(chain6HA, attr_divisor18);
```

And, then using epsilon relation defined above:

```
Relation omega1_divisor18 =
epsilon1_divisor18.leftResidual(epsilon1_divisor18).down();

Relation omega2_divisor18 =
epsilon2_divisor18.leftResidual(epsilon2_divisor18).down();
```

Generate the triangle operations: Two *triangle* operations of *Up* and *Down* can be defined for this example as below:

```
Relation upTriangle_divisor18 =
epsilon1_divisor18.starLeftResidual(I_divisor18).lambda();

Relation downTriangle_divisor18 =
epsilon2_divisor18.starLeftResidual(I_divisor18.transpose()).lambda();
```

Generate *i* and *c* versions: As already mentioned in the Chapters. 4. and 5., we can make all relations of *i* and *c* versions only based on *objects* (*extents*) or *attributes* (*intents*). We used only extents to generate all the mentioned relations in our implementation, i.g., *partial identities*.

```
Relation extents_divisor18 =
Relation.identity(chain6HA, epsilon2_divisor18.getCod())
.meet(downTriangle_divisor18.composition(upTriangle_divisor18));

Relation embExt_divisor18 =
extents_divisor18.emb("extents_divisor18");
```

i and *c* versions are used in generating *attribute implication* formulas described in Chapter. 4.
It is noteworthy that *c* versions display the relation that maps an intent to the corresponding extent (or display both Intents and Extents).

Generate attribute implications: As already mentioned in Chapters. 5.3.3 and 5.3.4, using 5 various versions of formulas all *attribute implications* can be generated (in this specific example only by extents). All formulas are built based on the generated relations defined in the previous steps.

Formula 1: $A_I = (i_M \,;\, \Omega^{\smile}) \setminus \Omega^{\smile}$

```
Relation AI1_divisor18 =
(extents_divisor18.composition(omega2_divisor18.transpose()))
.leftResidual(omega2_divisor18.transpose());
```

Formula 2: $A_I = \triangledown \, ; \triangle \, ; \Omega^{\smile}$

```
Relation AI2_divisor18 =
(downTriangle_divisor18.composition(upTriangle_divisor18))
.composition(omega2_divisor18.transpose());
```

Formula 3: $A_I = ((\mathbf{I} \,/\, \epsilon^{\smile}) \setminus (I/\epsilon^{\smile}))^{\downarrow}$

```
Relation AI3_divisor18 =
(I_divisor18.starRightResidual(epsilon2_divisor18.transpose()))
.leftResidual(I_divisor18.starRightResidual(epsilon2_divisor18
.transpose())).down();
```

Formula 4: $A_I = \triangledown \, ; \Omega \, ; \triangledown^{\smile}$

```
Relation AI4_divisor18 =
(downTriangle_divisor18.composition(omega1_divisor18))
.composition(downTriangle_divisor18.transpose());
```

Formula 5: $A_I = (c_M ; \Omega^{\smile}) \setminus (c_M ; \Omega^{\smile})$

```
Relation AI5_divisor18 =
(embExt_divisor18.composition(omega2_divisor18.transpose()))
.leftResidual(embExt_divisor18.composition(omega2_divisor1
.transpose()));
```

Show relations in the interface: following steps show how we can display our results (AI formulas) in a tab of our interface.

1. *RelationDisplay* class is generated to display all relations in our coding, i.e., attribute implications.

   ```
   RelationDisplay displayAI1_divisor18 = new RelationDisplay();
   displayAI1_divisor18.setRelation(AI1_divisor18);

   RelationDisplay displayAI2_divisor18 = new RelationDisplay();
   ```

```
displayAI2_divisor18.setRelation(AI2_divisor18);

RelationDisplay displayAI3_divisor18 = new RelationDisplay();
displayAI3_divisor18.setRelation(AI3_divisor18);

RelationDisplay displayAI4_divisor18 = new RelationDisplay();
displayAI4_divisor18.setRelation(AI4_divisor18);

RelationDisplay displayAI5_divisor18 = new RelationDisplay();
displayAI5_divisor18.setRelation(AI5_divisor18);
```

2. A new *JTabbedPane* is created to be displayed inside the Attribute Implications tab of the interface.

```
JTabbedPane tabbedPanel_divisor18 = new JTabbedPane();
```

3. A *JPanel* is created and added to the (*tabbedPane_divisor*18).

```
JPanel panel_divisor18 = new JPanel();
panel_divisor18.add(tabbedPanel_divisor18);
```

4. Since we prefer to show the attribute implications made by 5 different versions of AI formulas in various tabs of the interface, we have to add 5 *JPanel* for each tab.

```
JPanel panelAITab1_divisor18 = new JPanel();
panelAITab1_divisor18.add(displayAI1_divisor18);

JPanel panelAITab2_divisor18 = new JPanel();
panelAITab2_divisor18.add(displayAI2_divisor18);

JPanel panelAITab3_divisor18 = new JPanel();
panelAITab3_divisor18.add(displayAI3_divisor18);

JPanel panelAITab4_divisor18 = new JPanel();
panelAITab4_divisor18.add(displayAI4_divisor18);

JPanel panelAITab5_divisor18 = new JPanel();
panelAITab5_divisor18.add(displayAI5_divisor18);
```

5. Create and set up the main *JFrame*.

```
JFrame frame_Main = new JFrame("GUI");
frame_Main.setPreferredSize(new Dimension(1200, 600));
frame_Main.pack();
frame_Main.setLocationRelativeTo(null);
frame_Main.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame_Main.setVisible(true);
```

6. Add a *JTabbedPane* to the main *JFrame* to display several tabs vertically.

```
JTabbedPane tabbedPane_Main = new JTabbedPane();
tabbedPane_Main.setTabPlacement(JTabbedPane.LEFT);
frame_Main.add(tabbedPane_Main);
```

7. And finally, add *panel_divisor*18 to the *tabbedPane_Main* to display the attribute implications.

```
tabbedPane_Main.addTab(
"<html> <body leftmargin=15 topmargin=8 marginwidth=15
marginheight=5 color=\#348781> Attribute Implications
(star-based (3-valued) example) </body> </html>", panel_divisor18);
```

It is important to note that all other results shown in other tabs of the interface such as *context* and *HeytingAlgebra* can be shown easily using *OperationDisply* class. As we generated all these relations during this section, we only skip any explanations on their displays for their simplicity.

Chapter 6

Main Contributions Revisited

The principal purpose of this chapter is to go over the 4 main contributions represented in the Section. 1.1 and investigate how and where these contributions are achieved in this thesis. In the remainder of this chapter, we have used the same numbering as in Section. 1.1 so that comparison is straightforward.

1. As already mentioned in the Section. 4.1, Chapter. 4 introduces a *relational algebraic* approach to $\mathcal{L}$-fuzzy Concept Analysis. It is a *generalization* of regular Concept Analysis in the sense that for the category *Rel*, our approach is exactly the same as described in Chapter. 3. This is a proof of our claim about first main contribution. We expanded or generalized a regular Concept Analysis to generate $\mathcal{L}$-Fuzzy Concept Analysis using some new operations and terms relationally.
   In addition, our approach never tried to transfer the original input data into the crisp ones to reach this goal. It means that we worked with $\mathcal{L}$-Fuzzy Formal contexts directly. As previously stated in the Section. 5.3.1, we used some fuzzy contexts directly to proceed our work and generate our attribute implications.

2. In continuation of the first contribution of this thesis, our relational-algebraic framework generates in the Chapter. 5 using all preliminary and relational formulas and also FCA terms represented in the Chapters. 2, 4, and 3, respectively. Since all sections in the fifth chapter cover both regular and $\mathcal{L}$-Fuzzy ideas, it can be claimed that the second contribution of this thesis is also fulfilled. Its initial work started from the fourth chapter.

3. In the Figures. 5.4 and 5.5, all patterns or attribute implications are extracted from

the original input data. These attribute implications are calculated by 5 various versions of attribute implication formulas that we already obtained in Theorem. 4.2.1. As it is indicated by the mentioned figured, we tried to handle the ambiguity or uncertainty in the input data. i.e., the element values of matrices in these two figures are still values of $\mathcal{L}$ in Fuzzy idea. So, the third contribution of this thesis is covered as well.

4. All parts of our framework including main classes to define operations and classes for only representation purposes such as *RelationDisplay* are implemented by *Java* programming language. The consequences are shown in the figures of Chapter. 5. The most significant reason for choosing Java to implement our work is that it is a *OOP* (Object Oriented Programming) language.

Chapter 7

Conclusion and Future Work

This is the concluding chapter of the thesis as well as the future further works that can be applied to improve the process and result of this research.
This thesis is summarized in 5 main chapters that investigate the relational approach of $\mathcal{L}$-fuzzy Concept Analysis for the purpose of object classification based on the relation to their attributes. This procedure is fulfilled by considering the values in scale $\mathcal{L}$. These object-attribute classifications are performed to pursue the goal of limiting the ambiguity of object-attribute relationships in real data.

It is worthy to note that we considered both regular (2-valued) and fuzzy (3-valued) ideas in the implementation part of this research. This goes on further to prove that the proposed relation formulas in our framework can process both boolean and fuzzy context. So, the advantage of our approach is that we can work on some vague data without any need of manipulation to transform it to the non-vague data. To proof this, we started our implication with some $\mathcal{L}$-valued (not crisp) data as the data entry of our tables in the implementation.

The overall structure of the thesis and the general introduction of our work are reviewed in the first chapter. Later in other chapters, a detailed explanation of our approach is provided. Starting with a relation-algebraic representations of some helpful preliminaries for our research in the second chapter, in the chapter 3 we continue our work by FCA and Fuzzy-FCA (FFCA) definitions. Using a relational approach, we go further and deeper into the formulas we have covered so far in previous chapters and it leads to the creation of the attribute implications. In other words, the attribute implications are some patterns extracted from the data source. As we discussed earlier, in the case of fuzzy formal context there can be a large number of implications, which reduces the chance of exploring all im-

plications. So, we focus on only 5 versions of the attribute implications having the power to generate all other attribute implications.
Using this relation style, we implement our research by Java programming language and represent the goal visually in the final chapter.

Generally, time complexity is one of the most important factors in every research attempt. As a future extension of our research, a detailed investigation of the relations and formulas should be conducted to reduce the calculation time. This can be accomplished by optimizing formulas or perhaps initial classification of objects along with their related properties as input to save time. Also, another feature that can be added to our approach is to have input data without format restrictions such as image or text. It may add another phase to this work which is the proper format conversion to the relation before any performance.

Bibliography

[1] G. Addison, Izadpanahi A., Saha S., and M. Winter. $\mathcal{L}$-fuzzy concept analysis using fuzzy categories. Fuzzy Sets and Systems, 2022.

[2] G.T. Addison. Object classification using l-fuzzy concept analysis. MSc Thesis, Department of Computer Science, Brock University, Canada, 2020.

[3] M. S. da S. Alcântara, T. Dias, W. R. de Oliveira, and S. de B. Melo. A survey of categorical properties of l-fuzzy relations. In *Fuzzy sets and systems*, pages 62–82, 11 2021.

[4] G. Arevalo, S. Ducasse, and O. Nierstrasz. Lessons learned in applying formal concept analysis to reverse engineering. In *3rd International Conference on Formal Concept Analysis*, page 95–112, November 2005.

[5] W.W. Armstrong. Dependency structures in data base relationships. page 580–583. IFIP Congress, Geneva, Switzerland, 1974.

[6] A. Asperti and G. Longo. *Categories, types, and structures : an introduction to category theory for the working computer scientist.* Cambridge, Mass., : MIT Press, 1991.

[7] M. Barbut and B. Monjardet. *Odre et Classification, algebre et combinatoire*. Paris: Hachette, 1970.

[8] M.J. Benitez-Caballero, J. Medina, E. Ramirez-Poussa, and D. Slezak. Rough-set-driven approach for attribute reduction in fuzzy formal concept analysis. In *Fuzzy sets and systems*, pages 117–138, July 2020.

[9] R. Berghammer and Neumann F. RelView – an OBDD-based computer algebra system for relations. In *Gansha V.G., Mayr E.W., Vorozhtsov E. (eds.): Computer Algebra in Scientific Computing. LNCS, vol. 3718*, pages 40–51, 2005.

[10] R. Berghammer and M. Winter. Decomposition of relations and concept lattices. In *Fundamenta informaticae*, pages 37–82, 2013.

[11] R. Bělohlávek. Similarity relations in concept lattices. In *Journal of logic and computation*, pages 823–845. Oxford: Oxford University Press, 2000.

[12] R. Bělohlávek. What is a fuzzy concept lattice? ii. In *Kuznetsov S.O. et al (eds): Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC) 2011, LNAI 6743*, pages 19–26, 2011.

[13] R. Bělohlávek and J. Konecny. Scaling, granulation, and fuzzy attributes in formal concept analysis. page 1–6. IEEE, 2007.

[14] R. Bělohlávek and V. Vychodil. Attribute implications in a fuzzy setting.

[15] R. Bělohlávek and V. Vychodil. What is a fuzzy concept lattice? In *Proc. CLA 2005, Olomouc*, pages 34–45, 2005.

[16] P. Burmeister and R. Holzer. Treating incomplete knowledge in formal concept analysis. In *Formal Concept Analysis: Foundations and Applications*, pages 114–126, 01 2005.

[17] A. Burusco and R. Fuentes-Gonzalez. Construction of the l-fuzzy concept lattice. In *Fuzzy sets and systems*, pages 109–114, 1998.

[18] A. Buzmakov, E. Egho, N. Jay, S.O. Kuznetsov, A. Napoli, and C. Raïssi. On mining complex sequential data by means of fca and pattern structures. In *INTERNATIONAL JOURNAL OF GENERAL SYSTEMS, 2016*, pages 135–159, Feb 2016.

[19] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Elsevier, 2004, 2 edition, 1980.

[20] S. Cateni, V. Colla, and M. Vannucci. General purpose input variables extraction: A genetic algorithm based procedure give a gap. In *Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on*, pages 1278–1283, Nov 2009.

[21] R. Chadha, M. Viswanathan, and R. Viswanathan. Least upper bounds for probability measures and their applications to abstractions. In *Information and computation*, pages 68–106, 02 2014.

[22] J. Chi-Hyuck, S. Lee, H. Park, and J. Lee. Use of partial least squares regression for variable selection and quality prediction. In *Computers & Industrial Engineering (CIE 2009)*, 2009.

[23] L.H. Chin and A. Tarski. *Distributive and modular laws in the arithmetic of relation algebras, by Louise H. Chin and Alfred Tarski. Berkeley, University of California Press, 1951.* New York, Johnson Reprint Corp., 1966.

[24] L. Darniere. On the model-completion of heyting algebras. 2018.

[25] F. Dau and J. Klinger. From formal concept analysis to contextual logic. In *Formal Concept Analysis*, page 81–100, 01 2005.

[26] Y. Djouadi and H. Prade. Interval-valued fuzzy formal concept analysis. In *Foundations of Intelligent Systems*, pages 592–601, 2009.

[27] D. Doubois, W. Ostasiewicz, and H. Prade. Fuzzy sets: History and basic notions. Kluwer Academic Boston, 1999.

[28] P. Freyd and A. Scedrov. *Categories, allegories*. Amsterdam ; New York : North-Holland ; New York, NY, U.S.A. : Sole distributors for the U.S.A. and Canada, Elsevier Science Pub., 1990.

[29] G. Fu. Fca based ontology development for data integration. pages 765–782, 09 2016.

[30] H. Furusawa. A representation theorem for relation algebras: Concepts of scalar relations and point relations. In *Bulletin of informatics and cybernetics*, pages 109–119, March 1998.

[31] B. Ganter and R. Godin. Formal concept analysis third international conference, icfca 2005, lens, france, february 14-18, 2005 : proceedings. Berlin ; New York : Springer, 2005.

[32] B. Ganter and Wille R. Conceptual scaling. In *The IMA Volumes in Mathematics and Its Applications*, pages 139–167, 1989.

[33] B. Ganter, G. Stumme, and R. Wille. *Formal concept analysis : foundations and applications*, volume 3626. Berlin : Springer, 2005.

[34] B. Ganter and R. Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer-Verlag, 1st edn edition, 1999.

[35] D.E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison Wesley, 1989.

[36] G. Gratzer. The congruences of a finite lattice a proof-by-picture approach, third edition. In *Congruence lattices of finite lattices*, 2021.

[37] J.L. Guigues and V. Duquenne. Familles minimales d'implications informatives resultant d'un tableau de donn´ees binaires. page 5–18. Math. Sci. Humaines 95, 1986.

[38] T.P. Hong and Y.C. Lee. An overview of mining fuzzy association rules. pages 397–410. Fuzzy Sets and Their Extensions: Representation, Aggregation and Models, 2008.

[39] D.I. Ignatov. *Introduction to Formal Concept Analysis and Its Applications in Information Retrieval and Related Fields*. 2015.

[40] M. Jabri. Java programming language report. January 2021.

[41] A.B. Juandeaburre and R. Fuentes-González. The study of the l-fuzzy concept lattice. In *Mathware and Soft Computing*, pages 209–218, 1994.

[42] Y. Kawahara and H. Furusawa. Crispness and representation theorem in dedekind categories. 1998.

[43] Y. Kawahara and H. Furusawa. An algebraic formalization of fuzzy relations. In *Fuzzy Sets and Systems*, pages 125–135, January 1999.

[44] Ch.A. Kumar. Fuzzy clustering-based formal concept analysis for association rules mining. pages 274–301, 03 2012.

[45] H. Lai and D. Zhang. Concept lattices of fuzzy contexts: Formal concept analysis vs. rough set theory. In *International journal of approximate reasoning*, pages 695–707, 2009.

[46] A. M. Brito, L. C. de Barros, E. E. Laureano, F. M. Bertato, and M. E. Coniglio. Fuzzy formal concept analysis. In *Fuzzy Information Processing*, pages 192–205, 07 2018.

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

[48] C. Massri and F. Holik. On the representation of measures over bounded lattices. In *Algebra universalis*, 09 2021.

[49] J.P. Olivier and D. Serrato. Squares and rectangles in relation categories three cases: semilattice, distributive lattice and boolean non-unitary. In *Fuzzy sets and systems*, pages 167–187, June 1995.

[50] J. Poelmans, S.O. Kuznetsov, D.I. Ignatov, and G. Dedene. Formal concept analysis in knowledge processing: A survey on models and techniques. In *Expert systems with applications*, pages 6601–6623, November 2013.

[51] S. Pollandt. Datenanalyse mit fuzzy-begriffe. In *Stumme G., Wille R. (eds.): Begriffliche Wissensverarbeitung. Methoden und Anwendun- gen. Springer, Heidelberg*, pages 72–98, 2000.

[52] S. Prediger. Logical scaling in formal concept analysis. In *Conceptual Structures: Fulfilling Peirce's Dream*, pages 332–341, April 2006.

[53] G. Schmidt. *Relational mathematics*, volume 132. Cambridge, UK ; New York : Cambridge University Press, 2011.

[54] G. Schmidt and T. Strohlein. *Relations and graphs : discrete mathematics for computer scientists*. Berlin ; New York : Springer-Verlag, 1993.

[55] G.A. Tamargo, A. Bourget, and A. Pini. Discrete gauging and hasse diagrams. page 026, 08 2021.

[56] T. T.Quan, S.C. Hui, and T.H. Cao. A fuzzy fca-based approach to conceptual clustering for automatic generation of concept hierarchy on uncertainty data. In *the CLA 2004 International Workshop on Concept Lattices and their Applications, Ostrava, Czech Republic*, page 1–12, January 2004.

[57] R. Wille. *Restructuring lattice theory: An approach based on hierarchies of concepts*, volume Volume 83 of NATO Advanced Study Institutes Series. Springer Netherlands, ordered sets edition, 1982.

[58] R. Wille. Concept lattices and conceptual knowledge systems. In *Computers and mathematics with applications (1987)*, pages 493–515, 1992.

[59] M. Winter. A pseudo representation theorem for various categories of relations. In *Theory and Applications of Categories 7*, pages 23–37, March 2000.

[60] M. Winter. A new algebraic approach to l-fuzzy relations convenient to study crispness. In *Information sciences*, pages 233–252, December 2001.

[61] M. Winter. *Goguen Categories: A Categorical Approach to L-fuzzy Relations*, volume 25. Dordrecht: Springer Netherlands, 2007.

[62] M. Winter. Arrow categories. In *Fuzzy sets and systems*, pages 2893–2909, October 2009.

[63] M. Winter. Membership values in arrow categories. In *Fuzzy sets and systems*, pages 41–61, May 2015.

[64] M. Winter. Type-n arrow categories. In *Relational and Algebraic Methods in Computer Science (RAMiCS)*, pages 307–322, April 2017.

[65] M. Winter. T-norm based operations in arrow categories. In *Relational and Algebraic Methods in Computer Science*, pages 70–86, October 29 – November 1 2018.

[66] M. Winter and E. Jackson. Categories of relations for variable-basis fuzziness. In *Fuzzy Sets and Systems*, pages 222–237, September 2016.

[67] K.M. Yang, Kim E.H., S.H. Hwang, and S.H. Choi. Fuzzy concept mining based on formal concept analysis. In *INTERNATIONAL JOURNAL OF COMPUTERS*, 2008.

[68] K.M. Yang, E.H. Kim, Hwang S.H., and S.H. Choi. Conceptual analysis of fuzzy data using fca. 8th WSEAS International Conference on APPLIED COMPUTER SCIENCE, 2008.