Guy De Tré and Sławomir Zadrożny

The essence of fuzzy databases

June 8, 2015

Springer

Contents

Chapter 1
Introduction

In this short chapter we describe what is generally understood by imperfect information (section 1.1). After reading, the user should have insight in the different kinds of imperfections in information. Furthermore, we explain what is meant by ‘fuzzy’ databases (section 1.2), flexible querying and ‘fuzzy’ querying (section 1.3). Hereby we strive for general concept descriptions that are commonly acceptable. The last section 1.4 summarizes the objectives of this book.

1.1 What is meant by imperfect information?

A significant part of all information collected by humans is inherently **imperfect**. This imperfection follows from the way humans make use of natural language to communicate, think, behave and work. The imperfect character of information does not preclude us from successfully functioning in our society. For example, in order to drive a car, nobody needs perfect information that ‘the next side-way is exactly at 214,83 meter’, that ‘we should start using the brakes at exactly 1 minute and 43 seconds’, etc. It is just the human ability to make abstractions and to estimate for example time and distance that allows us perform complex tasks such as driving a car.

Imperfections in information can be classified as follows [17][1]:

- Information is *imprecise* if it is not specified as precise as it should be specified.
- Information is *fuzzy* if it is inherently vaguely described.
- Information is *uncertain* if it is not known with certainty.
- Information is *incomplete* if some data are missing.
- Information is *inconsistent* if there are two or more conflicting statements.

[1] Motro has presented a similar classification [16] where *incompleteness* is replaced by *ambiguity*: Information is said to be *ambiguous* if its meaning is not completely (clearly) given such that more interpretations and thus misinterpretations are possible.

In order to briefly describe these various forms of imperfection it is convenient to consider the information on the value of an attribute of some object. In what follows we will use the height of a person as an example. It will be treated as a broadly meant variable X.

1.1.1 Imprecision and vagueness

The concepts of *imprecision* and *vagueness* denote a restriction due to which information can only be described approximately. The inability to give an exact description can for example be due to round off errors in calculations, to a lack of necessary knowledge or to limitations of observation or measurement equipment.

If imperfect data are approximately modelled by precise boundaries like acceptable deviations or fault margins, then one speaks about **imprecision**. In this way the body length of a person could for example be given as ‘between 175 cm and 180 cm’ and the price of an object could be described as 500 ± 10 Euro. Such an imprecise information may be formally represented as “$X \in [175, 180]$”.

The concept **vagueness** is associated with the inability to describe information approximately within precise boundaries and therefore represents some kind of inherent imprecision. To handle vagueness, people will typically use linguistic terms to describe the inherent imprecision. Examples of vague descriptions for the height of a person are the linguistic terms ‘short’, ‘rather short’, ‘tall’ and ‘very tall’. Example of vague descriptions for price indications are the linguistic terms ‘very cheap’, ‘cheap’, ‘expensive’ and ‘unaffordable’. Such a vague information may be formally represented as “X IS *short*”.

From a semantic point of view, such linguistic terms have a *conjunctive* nature because they typically represent multiple candidate values. As such, there are for example multiple lengths which at the same time correspond with the term ‘short’. Moreover such linguistic terms are *inherently vague* concepts because not all candidates correspond to the same extent with the linguistic term. As such, a height of ‘200 cm’ corresponds to a larger extent with the term ‘tall’ than a height of ‘180 cm’.

1.1.2 Uncertainty

Like the concepts of *imprecision* and *vagueness*, the concept of **uncertainty** denotes a deficiency of information. Imprecise and vague information is not specific enough but is assumed to be accurate. We will explicitly refer to information as uncertain if the confidence in it is limited. An example of uncertain information is if John tells us that Jack’s height is 175 cm but we do not fully believe if John is telling us the truth. Such an uncertain information may be formally represented as “$X = 175$ is probable”.

There exists some affinity between the concepts of imprecision and vagueness on the one hand and the concept of uncertainty on the other hand: with each of them the exact information is not known. It is also very important to stress that imprecision and vagueness are **orthogonal** to uncertainty: they can occur independently next to each other or occur in combination.

An example can clarify this:

- The information on the height of a person can be imprecise: "$X \in [175, 180]$".
- The information can be uncertain: "$X = 175$ is probable".
- The information on the height can also be both imprecise and uncertain: "$X \in [175, 180]$ is probable"

Orthogonality concerns also vagueness and uncertainty:

- The information on the height of a person can be vague: "X IS tall".
- The information can be uncertain: "$X = 175$ is probable".
- The information on the height can also be both vague and uncertain: "X IS tall is probable"

Moreover the following relationship between imprecision/vagueness and uncertainty is often considered and is relevant for the topics dealt with in this book. Learning an imprecise/vague information on the value of a variable one is uncertain as to its actual value. The more imprecise the information the less certain one can be as to this actual value. This is the essence of the relationship between fuzzy sets theory and possibility theory which are briefly presented in the next chapter 2.

1.1.3 Incomplete and missing information

With the concept of **incompleteness** the missing of (a part of the) information is denoted. Information is **missing** if for some parts of it, no description is available, even not an imprecise, vague or uncertain description. Note that in case of imprecision, vagueness or uncertainty also some information is missing, namely the information that is necessary to give sufficient descriptions. However, in the remainder of this book and in the classification under consideration, the concept 'missing information' is used to denote only those cases where for the subject under consideration no information is available at all.

Missing information can be due to different causes. In the ANSI/X3/SPARC 'Interim Report' [1], fourteen different sources of incompleteness have been identified. In scientific research, these sources are usually reduced to the following five:

- Data *does not exists* or is *not applicable*. For example, the scores of an examination do not exist before the evaluation has been finished.
- *Unknown* data. The data exists, but is not known to the person who has to process it.
- *No* information. Nothing is known. It is possible that the data exist, but it is also possible that this is not the case.

- Data is known but *can not be entered*. For example, due to security reasons.
- Data are only *partially given*. The data are completely known, but are only partially entered in the system. For example, ABC codes for examination scores.

The fourth case of data that can not be entered will typically be handled by the database management system. In the fifth case of data that are only partially given, the data typically result from some aggregation process and can therefore be considered as derived data. Consequently, no extra modelling facilities are required for to deal with these cases. Therefore, in the remainder of this book only the first three cases are further dealt with.

1.1.4 Inconsistent information

Inconsistency (or ambiguity) describes a situation where two or more descriptions are conflicting, e.g. 'John is 1m 72' and 'John is larger than 1m 80'. In such cases, there is no possibility to combine the data in such a way that a compromise arises. A possible solution is to remove the information of the source that is least reliable (under the assumption that this source is known). Such an inconsistent information may be formally represented as, e.g., "$X = 175$ AND $X = 185$".

1.2 What are fuzzy databases?

When using regular computer software for the processing and management of information, users are almost always forced to describe data in a perfect way. This is because data modelling is usually done by means of data structures and data models that are developed to model information in a perfect way, taking into account the limitations of the computer system. Which is of course a direct consequence of the inherent binary nature of data storage and data processing.

Until some years ago the restrictions that come along with such an approach were not worth mentioning. The more because also for example in exact sciences like physics, we observe that people strive for certainty and preciseness and successfully make abstractions of reality to develop theories which are based on perfect information is an ideal world.

With the introduction of many-valued logics [2] (and later also [19]) a paradigm shift happened, which among others constitutes the onset for the so-called *fuzzy sets theory* [21], *possibility theory* [23, 9] and *fuzzy logic* [22]. These theories are nowadays sufficiently advanced to provide a very convenient and adequate framework for the formal handling of *imprecise*, *vague* and *uncertain* information. With the introduction of 'computing with words' [24, 26], the 'theory of fuzzy information granulation' [25] and the 'computational theory of perceptions" [27] the foundations are set to generalize all these theories into one single entirety.

Fuzzy sets theory is well suited for the modelling of *imprecise* and *vague* information, while *possibility theory* can be used for the handling of *uncertain* information and *incomplete* information. The basic concepts of these theories are therefore described in the next chapter 2. At the end of that chapter we also briefly introduce the basic ideas behind 'computing with words', 'fuzzy information granulation' and the 'theory of perceptions'.

Databases are a very important component of computer systems as these are the (structured) information sources for many applications. Traditional database management systems only allow to efficiently model and manage perfect information. The developments in fuzzy sets theory, possibility theory and fuzzy logic have also triggered research for advanced database models and database management techniques that additionally can deal with imperfect information. This results in the so-called **fuzzy databases** which intend to grasp imperfect information about a modelled part of the world and represent it directly in a database, preferably as natural as possible and preserving its semantics [4, 18, 7, 20, 3, 8, 15, 5].

Without fuzzy database techniques, one has to model imperfect information — which is a significant part of all available information— approximately in a crisp way. This very often goes hand in hand with simplification of semantics and thus a loss of information, which may be critical in some situations. Finding adequate solutions to the problem of imperfect information management has been identified by database researchers as one of the challenges for the near future [13].

1.3 What are flexible querying and fuzzy querying?

A widespread use of technologies for dealing with multimedia and large data collections (e.g., GIS databases and biological databases) has resulted in very large databases. Moreover, new developments in network and internet technology demand for distributed databases that are connected with each other and build up larger logical data sources. Because of the increasing number and volume of databases, good and accurate accessibility to a database becomes even more important. A lot of research has already been done to improve database access. In this research, many aspects have been dealt with, among which we mention file organization, indexing, querying techniques, query languages and other data access techniques.

Techniques that are meant to make database querying more flexible to users are generally called **flexible querying** techniques. These techniques include among others:

- Self-correcting querying systems that can correct syntactic and semantic errors in query formulations.
- Navigational querying systems that allow intelligent navigation through the database.
- Cooperative querying systems that support 'indirect' answers like summaries, conditional answers and contextual background information for (empty) results [10].

A specific kind of flexible querying techniques are based on fuzzy sets theory [21] and its related possibility theory [23, 9] and can therefore be called **fuzzy querying** techniques. In general fuzzy querying techniques aim to enhance database access by introducing fuzzy preferences in query formulations [6]. The introduction of fuzzy preferences in queries can be done at two levels: inside query conditions and between query conditions. Fuzzy preferences are introduced inside query conditions via flexible search criteria and allow to express that some values are more desirable than others in a gradual way. Fuzzy preferences between query conditions are expressed via grades of importance assigned to particular query conditions indicating that the satisfaction of some query conditions is more desirable than the satisfaction of others.

The research on fuzzy querying has already a long history. It has been inspired by the success of fuzzy logic in modelling natural language propositions. The use of such propositions in queries, in turn, seems to be very natural for human users of any information system, notably database management system. Later on, the interest in fuzzy querying has been reinforced by the omnipresence of network based applications, related to buzzwords of modern information technology, such as e-commerce, e-government, etc. These applications evidently call for a flexible querying capability when users are looking for some goods, hotel accommodations, etc., that may be best described using natural language terms like cheap, large, close to the airport, etc. Another amplification of the interest in fuzzy querying comes from developments in the area of data warehousing and data mining related applications. For example, a combination of fuzzy querying and data mining interfaces [11, 12] or fuzzy logic and the OLAP (Online Analytical Processing) technology [14] may lead to new, effective and more efficient solutions in this area.

1.4 What is this book about?

This book is meant as an introduction to the theory and practice of fuzzy querying and fuzzy databases. The book is intended for everybody who has a basic knowledge on database systems and can be used by master and PhD students as well as by researchers who want to become familiar with the use of ‘fuzzy’ techniques within information systems. The book is organized in such a way that the focus is on the techniques. As such, readers do not only learn about advanced database modelling and database access techniques, but also become familiar with the different ways how these techniques could be applied in other contexts.

References

1. ANSI/X3/SPARC, “Study Group on Data Base Management Systems: Interim Report”, *FDT ACM SIGMOD bulletin* **7** 2 (1975).
2. M. Black, “Vagueness: an exercise in logical analysis”, *Philosphy of Science* **4** 4 (1937) 427–455.
3. G. Bordogna, and G. Pasi (eds.), *Recent Issues on Fuzzy Databases* (Physica-Verlag, Heidelberg, Germany, 2000).
4. P. Bosc, and J. Kacprzyk (eds.), *Fuzziness in Database Management Systems* (Physica-Verlag, Heidelberg, Germany, 1995).
5. P. Bosc, L. Liétard, O. Pivert, and D. Rocacher, *Gradualité et imprécision dans les bases de données: ensembles flous, requêtes flexibles et interrogation de données mal connues* (Physica-Verlag, Heidelberg, Germany, 2005).
6. P. Bosc, D. Kraft, and F.E. Petry. “Fuzzy sets in database and information systems: status and opportunities”. *Fuzzy Sets and Systems* **153** 3 (2005) 418–426.
7. R. De Caluwe (ed.), *Fuzzy and Uncertain Object-Oriented Databases: Concepts and Models* (World Scientific, Signapore, 1997).
8. R. De Caluwe, G. De Tré, and G. Bordogna (eds.), *Spatio-Temporal Databases: Flexible Querying and Reasoning* (Physica, Heidelberg, Germany, 2004).
9. D. Dubois, and H. Prade, *Possibility Theory* (Plenum Press, New York, USA, 1988).
10. T. Gaasterland, P. Godfrey, and J. Minker, “An overview of cooperative answering”, *Journal of Intelligent Information Systems* **1** (1992) 123–157.
11. J. Kacprzyk, and S. Zadrożny, “On a fuzzy querying and data mining interface”, *Kybernetika* **36** (2000) 657–670.
12. J. Kacprzyk, and S. Zadrożny, “On combining intelligent querying and data mining using fuzzy logic concepts”, in: *Recent Research Issues on Fuzzy Databases*, G. Bordogna, and G. Pasi (eds.) (Physica-Verlag, Heidelberg, Germany, 2000) 67–81).
13. H.F. Korth, and A. Silberschatz, “Database Research Faces the Information Explosion”, *Communications of the ACM* **40** 2 (1997) 139–142.
14. A. Laurent, “Querying fuzzy multidimensional databases: unary operators and their properties”, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **11** (2003) 31–46.
15. Z. Ma (ed.), *Advances in Fuzzy Object-Oriented Databases: Modeling and Applications* (About Idea Group Inc, Hershey, USA, 2005).
16. A. Motro, “Management of Uncertainty in Database Systems”, in: *Modern Database Systems, The object model, interoperability and beyond*, W. Kim (ed.) (Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1995).
17. S. Parsons, “Current Approaches to Handling Imperfect Information in Data and Knowledge Bases”, *IEEE Transactions on Knowledge and Data Engineering* **8** 3 (1996) 353–372.

18. F.E. Petry, *Fuzzy Databases: Principles and Applications* (Kluwer Academic Publishers, Boston, USA, 1996).
19. N. Rescher, *Many-Valued Logic* (McGraw-Hill, New York, USA, 1969).
20. A. Yazici, and R. George, *Fuzzy Database Modeling* (Physica-Verlag, Heidelberg, Germany, 1999).
21. L.A. Zadeh, "Fuzzy Sets", *Information and Control* **8** 3 (1965) 338–353.
22. L.A. Zadeh, "Fuzzy logic and approximate reasoning", *Synthese* **30** 1 (1975) 407–428.
23. L.A. Zadeh, "Fuzzy sets as a basis for a theory of possibility", *Fuzzy Sets and Systems* **1** 1 (1978) 3–28.
24. L.A. Zadeh, "Fuzzy Logic = Computing with Words", *IEEE Transactions on Fuzzy Systems* **4** 2 (1996) 103–111.
25. L.A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic", *Fuzzy Sets and Systems* **90** 2 (1997) 111–127.
26. L.A. Zadeh, "Fuzzy Logic = Computing with Words", in: *Computing with Words in Information/Intelligent Systems*, Volume 1, L.A. Zadeh, and J. Kacprzyk (ed.) (Physica-Verlag, Heidelberg, Germany, 1999) 3–23.
27. L.A. Zadeh, "A New Direction in AI: Toward a Computational Theory of Perceptions", *AI Magazine* **22** 1 (2001) 73–84.

Chapter 2
Preliminaries

In this chapter we give an overview of the basic concepts and definitions of fuzzy set theory (section 2.1), possibility theory (section 2.2), and fuzzy logic (section 2.3). This overview should allow the user to understand the mathematics that are behind the ‘fuzzy’ database concepts that are described in this book. As this book deals mainly with the application of ‘fuzzy’ techniques, a lot of attention is paid to practical aspects of dealing with these theories. In section 2.4 we present the basics of a possibilistic logic that works with so-called possibilistic truth values and can be used as a logical framework for ‘fuzzy’ databases. Applications of this logic are presented throughout the remainder of the book. The chapter ends with an overview of some novel developments in fuzzy set theory which might be useful for the development of future ‘fuzzy’ database modelling and ‘fuzzy’ querying techniques (section 2.5).

2.1 Fuzzy set theory

Fuzzy set theory (also called the theory of fuzzy sets) is a generalization of classical set theory and was introduced in 1965 by L.A. Zadeh [22]. Since its introduction, this theory has been steadily developed and nowadays a lot of applications of it exist in several domains as for example informatics (logic programming, databases, data mining, artificial intelligence, knowledge-based systems, image processing, . . .), linguistics, medicine, sociology, psychology, geography, musicology, economics, etc. Fuzzy set theory is also at the basis of all the advanced database techniques described in this book.

In what follows some basic concepts and definitions of fuzzy set theory are presented. More detailed reference works are among others [9, 11, 7].

2.1.1 Definitions and notations

Central in fuzzy set theory is the concept *fuzzy set*. A fuzzy set is a generalization of the mathematical concept set.

To start with, we consider a universe of discourse U. Each mathematical set V of elements of U is fully characterized by its so-called **membership function** μ_V which associates the value 1 with each element of V and the value 0 with each other element of U, i.e.

$$\begin{aligned}\mu_V : U &\to \{0,1\}\\ x &\mapsto 1 \text{ iff } x \in V\\ x &\mapsto 0 \text{ iff } x \notin V\end{aligned}$$

Hereby, 1 is given the meaning 'belongs to the set', whereas 0 stands for 'does not belong to the set at all'.

For the definition of the concept fuzzy set, the discrete set $\{0,1\}$ of the previous mapping is extended to the unit interval $[0,1]$. The meaning of the value 0 remains the same, the value 1 is interpreted as '*fully* belongs to the fuzzy set' and an intermediate value $x \in]0,1[$ stands for 'only partially belongs (to an extent x) to the fuzzy set'. The closer the value x is to 0, the smaller the extent to which the associated element belongs to the fuzzy set.

Each **fuzzy set** is characterized by a (generalized) membership function. This is done as follows:

Definition 2.1 (Fuzzy set) *A fuzzy set $\tilde{V}$ over a universe of discourse U is defined by means of a (generalized) membership function $\mu_{\tilde{V}}$ which associates with each element x of U a membership grade $\mu_{\tilde{V}}(x) \in [0,1]$. This is done as follows:*

- $\mu_{\tilde{V}}(x) = 1$ *represents that x fully belongs to $\tilde{V}$ (with membership grade 1),*
- $\mu_{\tilde{V}}(x) = 0$ *represents that x does not belong to $\tilde{V}$ at all (and thus has a membership grade 0), and*
- $\mu_{\tilde{V}}(x) \in]0,1[$ *represents that x only partially belongs to $\tilde{V}$ (with membership grade $\mu_{\tilde{V}}(x)$).*

□

From the previous definition it follows straightforwardly that a set is a special case of a fuzzy set. Indeed, a set is a fuzzy set where each element that belongs to the fuzzy set, fully belongs to the fuzzy set (i.e. has an associated membership grade 1).

A fuzzy set $\tilde{V}$ over a universe of discourse U will in this work be generally denoted as

$$\tilde{V} = \{(x, \mu_{\tilde{V}}(x)) | \forall\, x \in U : \mu_{\tilde{V}}(x) > 0\}$$

By convention, elements with membership grade 0 are omitted in the notation. A fuzzy set $\tilde{V}$ with a finite number of elements will be denoted by

$$\tilde{V} = \{(x_1, \mu_{\tilde{V}}(x_1)), (x_2, \mu_{\tilde{V}}(x_2)), \ldots, (x_n, \mu_{\tilde{V}}(x_n))\}, n \in \mathbb{N}$$

where $\mathbb{N}$ represents the set of all natural numbers. The empty fuzzy set will also be represent by the symbol $\emptyset$.

Definition 2.2 (Fuzzy powerset $\tilde{\wp}(U)$) *The set of all fuzzy sets that can be defined over a universe of discourse U is itself defined by:*

$$\tilde{\wp}(U) \triangleq \{\tilde{V} | \tilde{V} \text{ satisfies definition 2.1}\}$$

and is called the fuzzy powerset of U. □

The membership function of a fuzzy set can be **discrete** or be **continuous** (cf. figure 2.1). In the context of 'fuzzy' databases, a fuzzy set is often used to model a linguistic term. As such, the fuzzy set which membership function is depicted on the left of figure 2.1 could be considered as a model for the linguistic term 'red touch', whereas the fuzzy set with the membership function on the right in the figure could be considered as a mathematical model for the term 'young (age)'. For such fuzzy sets, the linguistic term can be used as an *identifier*. With other words, in such a case the linguistic term identifies (the membership function of) the fuzzy set.

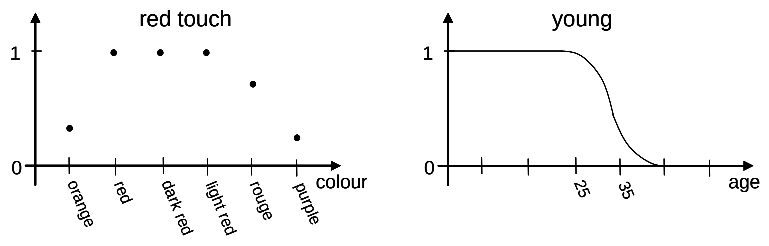

Fig. 2.1 Discrete and continuous membership functions.

2.1.2 Basic concepts

Among the most important concepts of fuzzy set theory are the concept **α-level set** (*α-cut*) and its variant **strict α-level set** (*strict α-cut*), that are defined as follows:

Definition 2.3 (α-level set and strict α-level set) *If $\tilde{V}$ is a fuzzy set which is defined over a universe of discourse U and α is a real number taken from the unit interval, i.e. $\alpha \in [0,1]$, then the α-level set $\tilde{V}_\alpha$ is by definition the (regular) set*

$$\tilde{V}_\alpha \triangleq \{x | x \in U \wedge \mu_{\tilde{V}}(x) \geq \alpha\}$$

and the strict α-level set $\tilde{V}_{\overline{\alpha}}$ is by definition the (regular) set

$$\tilde{V}_{\overline{\alpha}} \triangleq \{x | x \in U \wedge \mu_{\tilde{V}}(x) > \alpha\}$$

□

Two special cases of (strict) α-level sets of a fuzzy set $\tilde{V}$ are the **support** $\mathrm{supp}(\tilde{V})$ of the fuzzy set $\tilde{V}$ which is defined by:

$$\mathrm{supp}(\tilde{V}) \triangleq \{x | x \in U \wedge \mu_{\tilde{V}}(x) > 0\}$$

and the **core** $\mathrm{core}(\tilde{V})$ of the fuzzy set $\tilde{V}$ which is defined by:

$$\mathrm{core}(\tilde{V}) \triangleq \{x | x \in U \wedge \mu_{\tilde{V}}(x) = 1\}$$

A fuzzy set $\tilde{V}$ is **normalized** if its core $\mathrm{core}(\tilde{V})$ is a non-empty set, i.e. if $\mathrm{core}(\tilde{V}) \neq \emptyset$. If the core of fuzzy set is a singleton then the fuzzy set is called to be **unimodal**.

The **cardinality** of a fuzzy set is defined as follows:

Definition 2.4 (Cardinality) *Consider a fuzzy set $\tilde{V}$ that is defined over a universe of discourse U. If $\tilde{V}$ has a discrete membership function, then the cardinality* $\mathrm{card}(\tilde{V})$ *of $\tilde{V}$ is defined by:*

$$\mathrm{card}(\tilde{V}) \triangleq \sum_{x \in U} \mu_{\tilde{V}}(x)$$

If $\tilde{V}$ has a continuous membership function, then the cardinality $\mathrm{card}(\tilde{V})$ *of $\tilde{V}$ is defined by:*

$$\mathrm{card}(\tilde{V}) \triangleq \int_U \mu_{\tilde{V}}(x) dx$$

□

The cardinality of a fuzzy set results in a real number which reflects the global membership of all elements of the fuzzy set and can thus not be considered as an indication of the number of elements in the fuzzy set.

The **complement** of a fuzzy set is defined by [22]:

Definition 2.5 (Standard complement) *The (standard) complement $\overline{\tilde{V}}$ of a fuzzy set $\tilde{V}$ which is defined over a universe of discourse U is defined by:*

$$\overline{\tilde{V}} \triangleq \{(x, 1 - \mu_{\tilde{V}}(x)) | \forall\, x \in U : 1 - \mu_{\tilde{V}}(x) > 0\}$$

□

2.1.3 Fuzzy relations

A concept that is closely related to the concept fuzzy set is the concept fuzzy relation. A fuzzy relation is a generalization of the classic mathematical concept **relation**. Traditionally a relation

$$R : U_1 \times U_2 \times \cdots \times U_n \to Y$$

over a finite number of universa

$$U_1, U_2, \ldots, U_n, Y,\ n \in \mathbb{N} \setminus \{0\}$$

can be considered as being a subset of the Cartesian product

$$U_1 \times U_2 \times \cdots \times U_n \times Y.$$

Like with regular sets, this subset can be defined by means of a membership function μ_R which associates a value 1 with each $(n+1)$-tuple of $U_1 \times U_2 \times \cdots \times U_n \times Y$ that belongs to the relation R and associates a value 0 with all other $(n+1)$-tuples of $U_1 \times U_2 \times \cdots \times U_n \times Y$, i.e.

$$\begin{aligned}\mu_R : U_1 \times U_2 \times \cdots \times U_n \times Y &\to \{0,1\}\\ (x_1,x_2,\ldots,x_n,y) &\mapsto 1 \text{ iff } (x_1,x_2,\ldots,x_n,y) \in R\\ (x_1,x_2,\ldots,x_n,y) &\mapsto 0 \text{ iff } (x_1,x_2,\ldots,x_n,y) \notin R\end{aligned}$$

A **fuzzy relation** is then defined as follows:

Definition 2.6 (Fuzzy relation) *A fuzzy relation*

$$\tilde{R} : U_1 \times U_2 \times \cdots \times U_n \to Y$$

over a finite number of universa

$$U_1, U_2, \ldots, U_n, Y,\ n \in \mathbb{N} \setminus \{0\}$$

is defined by means of a (generalized) membership function $\mu_{\tilde{R}}$ which associates a membership grade

$$\mu_{\tilde{R}}((x_1,x_2,\ldots,x_n,y)) \in [0,1]$$

with each $(n+1)$-tuple $(x_1,x_2,\ldots,x_n,y)$ of the product set $U_1 \times U_2 \times \cdots \times U_n \times Y$. These membership grades have the following semantics:

- $\mu_{\tilde{R}}((x_1,x_2,\ldots,x_n,y)) = 1$ *means that $(x_1,x_2,\ldots,x_n,y)$ is a full element of $\tilde{R}$,*
- $\mu_{\tilde{R}}((x_1,x_2,\ldots,x_n,y)) = 0$ *means that $(x_1,x_2,\ldots,x_n,y)$ does not belong to $\tilde{R}$ and*
- $\mu_{\tilde{R}}((x_1,x_2,\ldots,x_n,y)) \in]0,1[$ *means that $(x_1,x_2,\ldots,x_n,y)$ only belongs to the given extent to $\tilde{R}$.*

□

A fuzzy relation

$$\tilde{R} : U_1 \times U_2 \times \cdots \times U_n \to Y$$

over a finite number of universa $U_1, U_2, \ldots, U_n, Y$, $n \in \mathbb{N} \setminus \{0\}$ will generally be denoted by

$$\begin{aligned}\tilde{R} = \{((x_1, x_2, \ldots, x_n, y), \mu_{\tilde{R}}((x_1, x_2, \ldots, x_n, y)))| \\ \forall\, (x_1, x_2, \ldots, x_n, y) \in U_1 \times U_2 \times \cdots \times U_n \times Y : \\ \mu_{\tilde{R}}((x_1, x_2, \ldots, x_n, y)) > 0\}\end{aligned}$$

Furthermore, from the previous definition it follows clearly that a classical relation can be seen as a special case of a fuzzy relation.

2.1.4 Operations

The semantics of a fuzzy set are completed by the definition of some operations. It is definitely not our intention to give a complete description of all operations in this subsection. For this purpose we refer to the literature, including among others [3, 12, 9, 7]. However, in this work we will pay attention to those operations that are relevant within the scope of 'fuzzy' database technology. More specifically we will deal with operations to compose fuzzy sets (union, intersection, difference and aggregation operators), operations to compare fuzzy sets (inclusion and equality), implication operators, quantifiers and extension principles for relations.

2.1.4.1 The composition of fuzzy sets

The traditional operations union, intersection and difference for regular sets could be generalised for fuzzy sets in several ways. Beside of that there exist some other operations to aggregate fuzzy sets. These allow to combine two or more fuzzy sets into one single fuzzy set.

Intersection and union.

The **standard definitions** for **intersection** and **union**, as originally presented by L.A. Zadeh [22], are as follows:

Definition 2.7 (Standard intersection and union) *With the understanding that $\tilde{V}_1$ and $\tilde{V}_2$ are two fuzzy sets that are defined over the same universe of discourse U, the following definitions hold:*

- *(Standard) intersection.*

$$\tilde{V}_1 \cap \tilde{V}_2 \triangleq \{(x, \min(\mu_{\tilde{V}_1}(x), \mu_{\tilde{V}_2}(x))) | \forall\, x \in U : \min(\mu_{\tilde{V}_1}(x), \mu_{\tilde{V}_2}(x)) > 0\}$$

- *(Standard) union.*

$$\tilde{V}_1 \cup \tilde{V}_2 \triangleq \{(x, \max(\mu_{\tilde{V}_1}(x), \mu_{\tilde{V}_2}(x))) | \forall\, x \in U : \max(\mu_{\tilde{V}_1}(x), \mu_{\tilde{V}_2}(x)) > 0\}$$

□

These definitions are only one of the many possibilities to define the intersection and union operations for fuzzy sets. Generally, the intersection and union operations are specified by means of a binary operation which is defined over the unit interval $[0,1]$ and satisfies some given conditions.

As such, the **intersection** of two fuzzy sets $\tilde{V}_1$ and $\tilde{V}_2$ —defined over the same universe of discourse U— can generally be specified by means of a function

$$i : [0,1] \times [0,1] \rightarrow [0,1]$$

which takes the membership grades of an element $x \in U$ in the fuzzy sets $\tilde{V}_1$ and $\tilde{V}_2$ as arguments and computes the membership grade of x in the intersection of $\tilde{V}_1$ and $\tilde{V}_2$, i.e.

$$\forall\, x \in U : \mu_{\tilde{V}_1 \cap \tilde{V}_2}(x) = i(\mu_{\tilde{V}_1}(x), \mu_{\tilde{V}_2}(x))$$

In order to be intuitively acceptable as an intersection function, the function i must moreover satisfy the following axioms:

- Axiom i1. $\forall\, a \in [0,1] : i(a,1) = a$ (border condition).
- Axiom i2. $\forall\, a,b,d \in [0,1] : b \leq d \Rightarrow i(a,b) \leq i(a,d)$ (monotonicity).
- Axiom i3. $\forall\, a,b \in [0,1] : i(a,b) = i(b,a)$ (commutativity).
- Axiom i4. $\forall\, a,b,d \in [0,1] : i(a,i(b,d)) = i(i(a,b),d)$ (associativity).

Functions i that satisfy the previous specification and axioms are in the literature known under the name **t-norms**. A such, each t-norm acts as an intersection operator. In table 2.1 we give some examples of (classes of) *t-norms*. For a more elaborated discussion and more examples we refer to [9].

Table 2.1: Examples of t-norms

Name	Formula $i(a,b)$	Parameter range
Zadeh [22]	$\min(a,b)$	
Lukasiewics	$\max(a+b-1,0)$	
Probabilistic	ab	
Yager [20]	$1-\min(1,[(1-a)^\omega+(1-b)^\omega]^{1/\omega})$	$\omega > 0$
Dubois [3]	$\dfrac{ab}{\max(a,b,\alpha)}$	$\alpha \in [0,1]$
Weber [19]	$\max(0, \dfrac{a+b+\lambda ab-1}{1+\lambda})$	$\lambda > -1$

The **union** of two fuzzy sets $\tilde{V}_1$ and $\tilde{V}_2$ —defined over the same universe of discourse U— can also generally be specified by means of a function

$$u : [0,1] \times [0,1] \to [0,1]$$

which takes the membership grades of an element $x \in U$ in the fuzzy sets $\tilde{V}_1$ and $\tilde{V}_2$ as arguments and computes the membership grade of x in the union of $\tilde{V}_1$ and $\tilde{V}_2$, i.e.

$$\forall\, x \in U : \mu_{\tilde{V}_1 \cup \tilde{V}_2} = u(\mu_{\tilde{V}_1}, \mu_{\tilde{V}_2})$$

In order to be intuitively acceptable as a union function, the function u must satisfy the following axioms:

- Axiom u1. $\forall\, a \in [0,1] : u(a,0) = a$ (border condition).
- Axiom u2. $\forall\, a,b,d \in [0,1] : b \leq d \Rightarrow u(a,b) \leq u(a,d)$ (monotonicity).
- Axiom u3. $\forall\, a,b \in [0,1] : u(a,b) = u(b,a)$ (commutativity).
- Axiom u4. $\forall\, a,b,d \in [0,1] : u(a,u(b,d)) = u(u(a,b),d)$ (associativity).

Functions u that satisfy the previous specification and axioms are in the literature known under the name **t-conorms**. As such, each t-conorm can act as a union operator.

For t-norms and t-conorms the following inequalities hold:

$$i(a,b) \leq a \leq u(a,b)$$
$$i(a,b) \leq b \leq u(a,b)$$

Furthermore it holds that

$$i(a,b) = 1 - u(1-a, 1-b)$$

which corresponds with the laws of de Morgan (i.e. $a \wedge b = \overline{\overline{a} \vee \overline{b}}$ and $a \vee b = \overline{\overline{a} \wedge \overline{b}}$). Because of this, for each t-norm there exists a corresponding t-conorm. In table 2.2 we give the corresponding (classes of) t-conorms for the (classes of) t-norms of table 2.1. For a more extensive discussion and more examples we refer to [9].

Table 2.2: Examples of t-conorms

Name	Formula $u(a,b)$	Parameter range
Zadeh [22]	$\max(a,b)$	
Lukasiewics	$\min(a+b,1)$	
Probabilistic	$a+b-ab$	
Yager [20]	$\min(1,(a^\omega+b^\omega)^{1/\omega})$	$\omega > 0$
Dubois [3]	$1-\dfrac{(1-a)(1-b)}{\max(1-a,1-b,\alpha)}$	$\alpha \in [0,1]$
Weber [19]	$\min(1,a+b-\dfrac{\lambda}{1-\lambda}ab)$	$\lambda > -1$

With respect to fuzzy databases, the Zadeh t-norm min and t-conorm max are frequently used because of their simple computability.

Set difference.

As is the case for regular sets the **difference** operator for fuzzy sets can be derived by relying on the definitions of t-norm and complement. Indeed

$$\mu_{\tilde{V}_1 \setminus \tilde{V}_2}(x) = \mu_{\tilde{V}_1 \cap \overline{\tilde{V}_2}}(x).$$

Definition 2.8 (Set difference) *With the understanding that $\tilde{V}_1$ and $\tilde{V}_2$ are two fuzzy sets that are defined on the same universe of discourse U, it holds that:*

$$\tilde{V}_1 \setminus \tilde{V}_2 = \{(x, i(\mu_{\tilde{V}_1}(x), 1 - \mu_{\tilde{V}_2}(x))) | \forall\, x \in U : i(\mu_{\tilde{V}_1}(x), 1 - \mu_{\tilde{V}_2}(x)) > 0\}$$

$\square$

This results for example with the Zadeh t-norm $\min(a,b)$ in

$$\mu_{\tilde{V}_1 \setminus \tilde{V}_2}(x) = \min(\mu_{\tilde{V}_1}(x), 1 - \mu_{\tilde{V}_2}(x))$$

and with the Lukasiewics t-norm $\max(a+b-1,0)$ in

$$\mu_{\tilde{V}_1 \setminus \tilde{V}_2}(x) = \max(\mu_{\tilde{V}_1}(x) - \mu_{\tilde{V}_2}(x), 0).$$

Aggregation operations.

An **aggregation operation** is used to combine two or more fuzzy sets in a desirable way into one single fuzzy set. An aggregation operation over a finite number of fuzzy sets $\tilde{V}_1, \tilde{V}_2, \ldots, \tilde{V}_n$, $n \in \mathbb{N} \setminus \{0\}$ —all defined over the same universe of discourse U— can generally be specified by means of a function

$$h : [0,1]^n \to [0,1]$$

which takes the membership grades of an element $x \in U$ in the fuzzy sets $\tilde{V}_1, \tilde{V}_2, \ldots, \tilde{V}_n$ as arguments and returns the membership grade of x in the fuzzy set $\tilde{V}$ of the aggregate, i.e.

$$\forall\, x \in U : \mu_{\tilde{V}}(x) = h(\mu_{\tilde{V}_1}(x), \mu_{\tilde{V}_2}(x), \ldots, \mu_{\tilde{V}_n}(x))$$

In order to be a meaningful aggregation operator, such a function h minimally has to satisfy the following axioms:

- Axiom h1. $h(0,0,\ldots,0) = 0 \wedge h(1,1,\ldots,1) = 1$ (border conditions).
- Axiom h2.

$$\forall\,(a_1,a_2,\dots,a_n),(b_1,b_2,\dots,b_n)\in[0,1]^n,\forall\,i\in\{1,2,\dots,n\}:$$
$$a_i\le b_i\Rightarrow h(a_1,a_2,\dots,a_n)\le h(b_1,b_2,\dots,b_n)$$

(h is non-decreasing in all of its arguments).

If an aggregation operator satisfies the following axioms h3, h4 or h5, the operator is respectively said to be continuous, symmetric or idempotent.

- Axiom h3. h is a continuous function (continuity).
- Axiom h4. For each permutation p of $\{1,2,\dots,n\}$ it must hold that

$$h(a_1,a_2,\dots,a_n)=h(a_{p(1)},a_{p(2)},\dots,a_{p(n)})$$

  (h is a symmetric function in all of its arguments).
- Axiom h5. $\forall\,a\in[0,1]:h(a,a,\dots,a)=a$ (idempotency).

Examples of (classes of) aggregation operators are the so-called generalized averages and the so-called ordered weighted averages (OWAs).

The **generalized averages** are defined by

$$h(a_1,a_2,\dots,a_n;\alpha)\triangleq\left(\frac{a_1^\alpha+a_2^\alpha+\dots+a_n^\alpha}{n}\right)^{1/\alpha}$$

where $\alpha\in\mathbb{R}_0$ is the parameter that distinguishes the different aggregation operators and it must hold that $a_i\neq 0$, $i=1,2,\dots,n$ if $\alpha<0$.

The **ordered weighted averages** are defined by

$$h(a_1,a_2,\dots,a_n;w)\triangleq w_1b_1+w_2b_2+\dots+w_nb_n$$

where $w=(w_1,w_2,\dots,w_n)\in[0,1]^n$ is called the weighting factor. It must hold that $\sum_{i=1}^n w_i=1$ and that for each $i\in\{1,2,\dots,n\}$, b_i is the i^{th} largest element of $a_1,a_2,\dots,a_n$, i.e. $(b_1,b_2,\dots,b_n)$ is a permutation of $(a_1,a_2,\dots,a_n)$ where the elements are ordered as follows:

$$\forall\,i,j\in\{1,2,\dots,n\}:\ i<j\Rightarrow b_i\ge b_j.$$

2.1.4.2 The comparison of fuzzy sets

When comparing regular sets, the result is always expressed by means of a classical Boolean truth value 'true' (T or 1) or 'false' (F or 0). For the comparison of fuzzy sets two approaches are used: a standard approach where the result is a Boolean truth value and a gradual approach where result of the comparison is expressed by a real number between 0 and 1. The latter approach is more flexible and allows to consider gradations of truth.

Inclusion.

For regular sets V_1 and V_2 over a universe of discourse U the inclusion can be expressed with the help of the implication operator of the Boolean logic. This can be done as follows:

$$(V_1 \subseteq V_2) \Leftrightarrow (\forall\, x \in U : (x \in V_1) \Rightarrow (x \in V_2))$$

For fuzzy sets the **standard definitions** for **inclusion** are based on the same principle and given as follows [22].

Definition 2.9 (Inclusion and strong inclusion) *With the understanding that $\tilde{V}_1$ and $\tilde{V}_2$ are two fuzzy sets that are defined over the same universe of discourse U, it holds that:*

- *(Standard) inclusion.* $(\tilde{V}_1 \subseteq \tilde{V}_2) \Leftrightarrow (\forall\, x \in U : \mu_{\tilde{V}_1}(x) \leq \mu_{\tilde{V}_2}(x))$
- *(Standard) strong inclusion.* $(\tilde{V}_1 \sqsubseteq \tilde{V}_2) \Leftrightarrow \mathrm{supp}(\tilde{V}_1) \subseteq \mathrm{core}(\tilde{V}_2)$

□

The standard inclusion is illustrated in figure 2.2: in figure (a) $\tilde{V}_1 \subseteq \tilde{V}_2$ is satisfied; in figure (b) $\tilde{V}_1 \subseteq \tilde{V}_2$ is not satisfied.

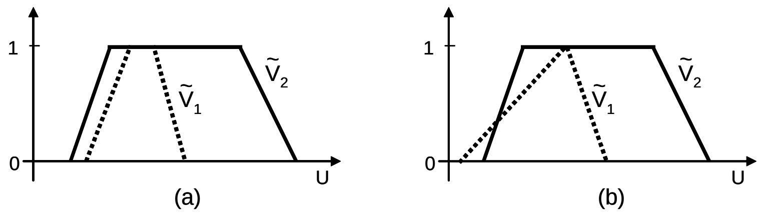

Fig. 2.2 (Standard) inclusion.

With the standard strong inclusion $\tilde{V}_1 \sqsubseteq \tilde{V}_2$ it is expressed that if an element belongs to the support of $\tilde{V}_1$, then this element must necessarily belong to the core of $\tilde{V}_2$. This is illustrated in figure 2.3: it holds that $\tilde{V}_1 \sqsubseteq \tilde{V}_2$.

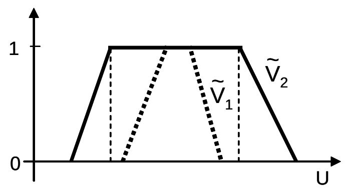

Fig. 2.3 (Standard) strong inclusion.

Considering the **gradual** approach, multiple definitions exist. A frequently used method to find the truth value of $\tilde{V}_1 \subseteq \tilde{V}_2$ is to check to which extent the fuzzy set $\tilde{V}_1$ 'matches' the fuzzy set $\tilde{V}_1 \cap \tilde{V}_2$. Hereby, the following considerations with respect to inclusion of regular sets are used:

$$(V_1 \subseteq V_2) \Leftrightarrow ((V_1 \cap V_2) = V_1)$$

or

$$(V_1 \subseteq V_2) \Leftrightarrow (\frac{\text{card}(V_1 \cap V_2)}{\text{card}(V_1)} = 1)$$

which leads to the following definition for the truth value $\deg(\tilde{V}_1 \subseteq \tilde{V}_2)$ of $\tilde{V}_1 \subseteq \tilde{V}_2$.

Definition 2.10 (Gradual inclusion) *With the understanding that $\tilde{V}_1$ and $\tilde{V}_2$ are two fuzzy sets that are defined over the same universe of discourse U, it holds that:*

$$\deg(\tilde{V}_1 \subseteq \tilde{V}_2) = \frac{\text{card}(\tilde{V}_1 \cap \tilde{V}_2)}{\text{card}(\tilde{V}_1)}$$

□

For fuzzy sets with a discrete membership function the previous definition results in

$$\deg(\tilde{V}_1 \subseteq \tilde{V}_2) = \frac{\sum_{x \in U} i(\mu_{\tilde{V}_1}(x), \mu_{\tilde{V}_2}(x))}{\sum_{x \in U} \mu_{\tilde{V}_1}(x)}$$

where i is a t-norm.

For fuzzy sets with a continuous membership function the definition becomes

$$\deg(\tilde{V}_1 \subseteq \tilde{V}_2) = \frac{\int_U i(\mu_{\tilde{V}_1}(x), \mu_{\tilde{V}_2}(x))dx}{\int_U \mu_{\tilde{V}_1}(x)dx}$$

where i is again a t-norm.

Equality.

The **standard definition** for the **equality** of two fuzzy sets which are both defined over a universe of discourse U is given as follows [22].

Definition 2.11 (Standard equality) *With the understanding that $\tilde{V}_1$ and $\tilde{V}_2$ are two fuzzy sets that are defined over the same universe of discourse U, it holds that:*

$$(\tilde{V}_1 = \tilde{V}_2) \Leftrightarrow (\forall\, x \in U : \mu_{\tilde{V}_1}(x) = \mu_{\tilde{V}_2}(x))$$

□

A frequently used **gradual** approach is based on the fact that for regular sets V_1 and V_2 over a universe of discourse U equality can be expressed as follows:

$$(V_1 = V_2) \Leftrightarrow ((V_1 \subseteq V_2) \wedge (V_2 \subseteq V_1))$$

Analogously as with gradual inclusion we then obtain the following definition for the truth value $\deg(\tilde{V}_1 = \tilde{V}_2)$ of $\tilde{V}_1 = \tilde{V}_2$.

Definition 2.12 (Gradual equality) *With the understanding that $\tilde{V}_1$ and $\tilde{V}_2$ are two fuzzy sets that are defined over the same universe of discourse U, it holds that:*

$$\deg(\tilde{V}_1 = \tilde{V}_2) = i(\deg(\tilde{V}_1 \subseteq \tilde{V}_2), \deg(\tilde{V}_2 \subseteq \tilde{V}_1))$$

where i is a t-norm. $\square$

2.1.4.3 Implication operators

The implication operator of Boolean logic has already been mentioned with the definition of inclusion. In essence the traditional implication $(P \Rightarrow Q)$ —if P, then Q— is equivalent to the logical expression $((\neg P) \vee Q)$. In a fuzzy framework, the propositions P and Q no longer take Boolean truth values, but more generally take values from the unit interval $[0,1]$. Like with fuzzy intersection and fuzzy union, a **fuzzy implication operator** $\Rightarrow_f$ can generally be defined by a function:

$$\begin{aligned} \Rightarrow_f \colon [0,1] \times [0,1] &\rightarrow [0,1] \\ (p,q) &\mapsto (p \Rightarrow_f q) \end{aligned}$$

The more the function values are closer to 1 (resp. 0), the more the result of the implication operator is true (resp. false). As with t-norms and with t-conorms, there exist multiple families of fuzzy implication operators. In what follows, three such families which are frequently used within the context of 'fuzzy' databases, are described: S-implications, R-implications and contrapositions of R-implications.

S-implications.

The name **S-implication** stems from the fact that these kinds of implication operations are based on a t-conorm. In the literature, a t-conorm is often denoted with the symbol S. The definition of S-implications is based on the expression $(P \Rightarrow Q) \Leftrightarrow ((\neg P) \vee Q)$.

Definition 2.13 (S-implication) *With P and Q two fuzzy propositions, the family of S-implications (denoted by $\Rightarrow_{S-i}$) is defined by:*

$$(P \Rightarrow_{S-i} Q) = u(1 - p, q)$$

where u is a t-conorm. $\square$

The most important S-implication functions are given in table 2.3.

Table 2.3: Examples of S-implications

Name	Definition	Used t-conorm
Kleene-Dienes	$(P \Rightarrow_{K-D} Q) = \max(1-p, q)$	$u(a,b) = \max(a,b)$
Lukasiewics	$(P \Rightarrow_{LuS} Q) = \min(1-p+q, 1)$	$u(a,b) = \min(a+b, 1)$
Reichenbach	$(P \Rightarrow_{Rb} Q) = 1-p+p.q$	$u(a,b) = a+b-a.b$

As contraposition of an S-implication we receive the same S-implication because:

$$\begin{aligned}(\neg Q \Rightarrow_{S-i} \neg P) &= u(1-(1-q), 1-p) = u(q, 1-p) = u(1-p, q)\\ &= (P \Rightarrow_{S-i} Q)\end{aligned}$$

Furthermore, it can be proved that [9]

$$(P \Rightarrow_{K-D} Q) \leq (P \Rightarrow_{Rb} Q) \leq (P \Rightarrow_{Lu} Q)$$

The largest S-implication is obtained by using the t-conorm of Weber:

$$(P \Rightarrow_{We} Q) = \begin{cases} 1-p & \text{iff } q = 0\\ q & \text{iff } p = 1\\ 1 & \text{else.}\end{cases}$$

R-implications.

The family of **R-implications** is named in this way because their definition is based on the principle of *residuation*, which is traditionally described as follows:

$$(P \wedge (P \Rightarrow Q)) \Leftrightarrow (P \wedge (\neg P \vee Q)) \Leftrightarrow ((P \wedge \neg P) \vee (P \wedge Q)) \Leftrightarrow (P \wedge Q)$$

from which it follows that

$$(p \wedge (p \Rightarrow q)) \leq q$$

This leads to the following definition.

Definition 2.14 (R-implication) *With P and Q two fuzzy propositions, the family of R-implications (denoted by $\Rightarrow_{R-i}$) is defined by:*

$$(P \Rightarrow_{R-i} Q) = \sup_{x \in [0,1]} \{x | i(p,x) \leq q\}$$

where i is a t-norm. □

An equivalent, more frequently used form for R-implications is:

$$(P \Rightarrow_{R-i} Q) = \begin{cases} 1 & \text{iff } p \leq q \\ f(p,q) & \text{else.} \end{cases}$$

where f is the function that characterizes the R-implication. The latter follows from definition 2.14 and the fact that:

$$(\forall\, x \in [0,1] : i(x,p) \leq i(1,p))$$

such that

$$(\forall\, x \in [0,1] : i(x,p) \leq p)$$

If $p \leq q$ one has thus:

$$(\forall\, x \in [0,1] : i(x,p) \leq p \leq q)$$

Because we search for the largest $x \in [0,1]$ for which the inequality holds, we obtain the value 1, what guarantees the alternative form:

$$(P \Rightarrow_{R-i} Q) = 1 \text{ iff } p \leq q$$

A special implication which can due to its form can be considered as being an R-implication, is the implication of Rescher-Gaines

$$(P \Rightarrow_{R-G} Q) = \begin{cases} 1 & \text{iff } p \leq q \\ 0 & \text{else.} \end{cases}$$

This implication always returns a Boolean value.

In table 2.4 we represent the most important R-implication functions.

Table 2.4: Examples of R-implications

Name	Definition	Used t-norm
Gödel	$(P \Rightarrow_{Go} Q) = \begin{cases} 1 & \text{iff } p \leq q \\ q & \text{else} \end{cases}$	$i(a,b) = \min(a,b)$
Lukasiewicz	$(P \Rightarrow_{LuR} Q) = \begin{cases} 1 & \text{iff } p \leq q \\ 1-p+q & \text{else} \end{cases}$	$i(a,b) = \max(a+b-1,0)$
Goguen	$(P \Rightarrow_{Gg} Q) = \begin{cases} 1 & \text{iff } p \leq q \\ q/p & \text{else} \end{cases}$	$i(a,b) = ab$

It cab be proved that [9]

$$(P \Rightarrow_{R-G} Q) \leq (P \Rightarrow_{Go} Q) \leq (P \Rightarrow_{Gg} Q) \leq (P \Rightarrow_{lu} Q)$$

Contrapositions of R-implications.

A third family concerns the implications that *correspond* to the **contrapositions of R-implications**. We hereby depart from the definition:

$$(P \Rightarrow_{cont\ R-i} Q) = ((\neg Q) \Rightarrow_{R-i} (\neg P))$$

For the Gödel R-implication this results in the following definition

$$(P \Rightarrow_{cont\ Go} Q) = \begin{cases} 1 & \text{iff } p \leq q \\ 1-p & \text{else} \end{cases}$$

For the Goguen R-implication we obtain

$$(P \Rightarrow_{cont\ Gg} Q) = \begin{cases} 1 & \text{iff } p \leq q \\ \dfrac{1-p}{1-q} & \text{else} \end{cases}$$

For the Lukasiewics R-implication and the Rescher-Gaines R-implication is the contraposition the same as the original R-implication. Indeed,

$$\begin{aligned}(P \Rightarrow_{cont\ Lu} Q) &= ((\neg Q) \Rightarrow_{Lu} (\neg P)) \\ &= \begin{cases} 1 & \text{iff } (1-q) \leq (1-p) \\ 1-(1-q)+(1-p) & \text{else} \end{cases} \\ &= \begin{cases} 1 & \text{iff } p \leq q \\ 1-p+q & \text{else} \end{cases} \\ &= (P \Rightarrow_{Lu} Q)\end{aligned}$$

and

$$\begin{aligned}(P \Rightarrow_{cont\ R-G} Q) &= ((\neg Q) \Rightarrow_{R-G} (\neg P)) \\ &= \begin{cases} 1 & \text{iff } (1-q) \leq (1-p) \\ 0 & \text{else} \end{cases} \\ &= \begin{cases} 1 & \text{iff } p \leq q \\ 0 & \text{else} \end{cases} \\ &= (P \Rightarrow_{R-G} Q)\end{aligned}$$

2.1.4.4 Quantifiers

Fuzzy sets allow it to define a broad range of quantifiers [27]. Indeed, beside the universal quantifier ($\forall$) and the existential quantifier ($\exists$), we can consider quantifiers that are described by linguistic terms. A distinction is made between **absolute quantifiers** which denote a number or quantity like, e.g., ‘around twelve’, ‘around six’, etc. and **relative quantifiers** which refer to a total number and denote a proportion of this total like, e.g. ‘most’, ‘a small number’, etc.

An absolute quantifier Q_A is modelled by means of a membership function

$$\mu_{Q_A} : \mathbb{N} \to [0,1] \text{ of } \mu_{Q_A} : \mathbb{R} \to [0,1]$$

A relative quantifier by a membership function

$$\mu_{Q_R} : [0,1] \to [0,1]$$

Hereby, the value $\mu_{Q_A}(n)$ expresses the extent to which the number n corresponds to the quantifier; analogously, the value $\mu_{Q_R}(p)$ expresses the extent to which the proportion p corresponds to the quantifier.

Quantifiers play an important role in fuzzy queries where it is required that ‘Q X satisfy C’, where Q is a quantifier, C is a condition and X represents a collection of elements. The intention is then to determine to which extent it is the case that Q elements of X satisfy C, or with other words to determine to which extent the number of elements (resp. the proportion) of X that satisfy C is compatible with Q.

2.1.4.5 Extension principles for relations

A (regular) relation

$$R : U_1 \times U_2 \times \cdots \times U_n \to Y$$

which is defined over a finite number of universa

$$U_1, U_2, \ldots, U_n, Y,\ n \in \mathbb{N} \setminus \{0\}$$

can be generalized to a **generalized relation**

$$\tilde{R} : \tilde{\wp}(U_1) \times \tilde{\wp}(U_2) \times \cdots \times \tilde{\wp}(U_n) \to \tilde{\wp}(Y)$$

which acts on fuzzy sets that are defined over the universa $U_1, U_2, \ldots, U_n$ and results in a fuzzy set that is defined over the universe of discourse Y.

A method to obtain such a generalized relation is called an **extension principle**. One of the best known extension principles is the extension principle of L.A. Zadeh [23, 24, 25] which is defined as follows:

Definition 2.15 (Extension principle) *With the understanding that $U_1, U_2, \ldots, U_n$ and Y are universa of discourse $(n \in \mathbb{N} \setminus \{0\})$ and that*

$$R : U_1 \times U_2 \times \cdots \times U_n \to Y$$

is a regular relation, the generalized relation $\tilde{R}$ of R is defined by:

$$\begin{aligned}\tilde{R}: \tilde{\wp}(U_1) \times \tilde{\wp}(U_2) \times \cdots \times \tilde{\wp}(U_n) &\to \tilde{\wp}(Y)\\ \tilde{V}_1, \tilde{V}_2, \ldots, \tilde{V}_n &\mapsto \tilde{R}(\tilde{V}_1, \tilde{V}_2, \ldots, \tilde{V}_n)\end{aligned}$$

where $\tilde{R}(\tilde{V}_1, \tilde{V}_2, \ldots, \tilde{V}_n)$ is the fuzzy set over the universe of discourse Y that is defined by:

$$\begin{aligned}&\tilde{R}(\tilde{V}_1,\tilde{V}_2, \ldots, \tilde{V}_n): Y \to [0,1]\\ &\quad y \mapsto \sup_{(x_1,x_2,\ldots,x_n) \in V_{(R,y)}} \min(\mu_{\tilde{V}_1}(x_1), \mu_{\tilde{V}_2}(x_2), \ldots, \mu_{\tilde{V}_n}(x_n)), \forall\, y \in range(R)\\ &\quad y \mapsto 0, \forall\, y \notin range(R)\end{aligned}$$

where $range(R)$ is the value set of the relation R, i.e.

$$range(R) = \{y | y \in Y \wedge \exists\, (x_1, x_2, \ldots, x_n) \in U_1 \times U_2 \times \cdots \times U_n : R(x_1, x_2, \ldots, x_n) = y\}$$

and $V_{(R,y)}$ is the set of all n-tuples which mapped onto the value y by means of the relation R, i.e.

$$V_{(R,y)} = \{(x_1, x_2, \ldots, x_n) | (x_1, \ldots, x_n) \in U_1 \times U_2 \times \cdots \times U_n \wedge R(x_1, x_2, \ldots, x_n) = y\}$$

□

In the previous definition of the extension principle of L.A. Zadeh the Zadeh t-norm (cf. table 2.1) is used. By using other t-norms, other extension priciples can be defined. An example is the extension principle of R.R. Yager [21] where the Yager t-norm (cf. table 2.1) is used.

With respect to fuzzy databases, extension principles can be used to generalize operators of regular data types. In such a way, the addition operator $+$ of the data type *Integer*, which is used for the modelling of integer values, can for example be generalized to an addition operator $\tilde{+}$ which acts on two fuzzy sets and results in a fuzzy set that are all defined over the universe of discourse $\mathbb{Z}$ of integer numbers. A major disadvantage is that in practice, the use of an extension principle is in most cases very difficult to implement in software.

2.1.5 Shape functions and practical issues

A continuous membership function can be approached by a so-called **shape function**. This might be very useful for practical modelling purposes for software implementations. Frequently used shape functions are: the S-function, the π-function and the Π-function. Other shape functions, that are important in practice are the so-called trapezoidal and triangular membership functions. Each of these shape functions is further described below. In figure 2.4 some examples of them are illustrated.

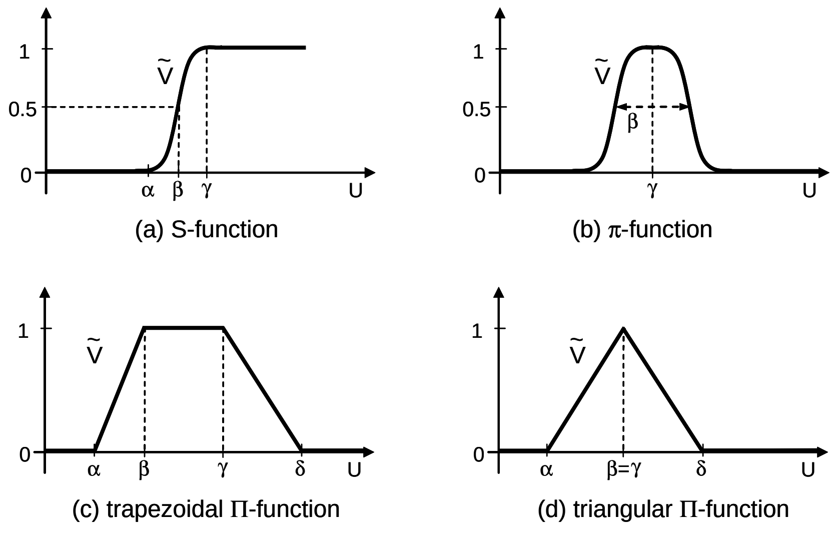

Fig. 2.4 Some examples of shape functions.

2.1.5.1 The *S*-function

This function is characterized by two parameter values α and γ which are both elements of the set of real numbers $\mathbb{R}$, i.e. $(\alpha,\gamma) \in \mathbb{R}^2$. Furthermore, it holds that:

$$\alpha < \gamma$$

Domain values that are strict smaller than α are mapped onto the function value 0. Domain values which are strict larger than γ are mapped onto the function value 1. A special domain value

$$\beta = \frac{\alpha+\gamma}{2}$$

has 0.5 as function value and is therefore called the ‘transition point’. For all other domain values —that are element of the $[\alpha,\gamma]$— the function value is obtained by a quadratic interpolation (see example (*a*) in figure 2.4). For each $(\alpha,\gamma) \in \mathbb{R}^2 : \alpha < \gamma$ and $\beta = \dfrac{\alpha+\gamma}{2}$, the *S*-function is formally defined by:

$$S(.;\alpha,\gamma) : U \to [0,1]$$
$$x \mapsto \begin{cases} 0 & \text{iff } x \in]-\infty,\alpha[\\ 2(\dfrac{x-\alpha}{\gamma-\alpha})^2 & \text{iff } x \in [\alpha,\beta] \\ 1-2(\dfrac{x-\gamma}{\gamma-\alpha})^2 & \text{iff } x \in]\beta,\gamma] \\ 1 & \text{iff } x \in]\gamma,\infty[\end{cases}$$

2.1.5.2 The π-function

The π-function is a clock function which is constructed by ‘concatenating’ an S-function with its mirrored image. Hereby two parameter values $(\beta,\gamma) \in \mathbb{R}^2$ must be provided. The parameter β is called the bandwidth and determines the distance between the two transition points. Therefore it is required that $\beta > 0$. The parameter γ denotes the domain value of the top (maximum) of the clock function (see example (b) in figure 2.4). For each couple $(\beta,\gamma) \in \mathbb{R}^2$ with $beta > 0$, the π-function is formally defined by:

$$\pi(.;\beta,\gamma) : U \to [0,1]$$
$$x \mapsto \begin{cases} S(x;\gamma-\beta,\gamma) & \text{iff } x \in]-\infty,\gamma[\\ 1-S(x;\gamma,\gamma+\beta) & \text{iff } x \in [\gamma,\infty[\end{cases}$$

2.1.5.3 The Π-function

This is a general shape function which is constructed using two functions π_1 and π_3 and four parameter values $(\alpha,\beta,\gamma,\delta) \in \mathbb{R}^4$.

The functions π_1 and π_3 must both be continuous mappings from the unit interval $[0,1]$ onto itself, for which the following border conditions must hold:

$$\begin{cases} \pi_1(0) = 0 \text{ and } \pi_1(1) = 1 \\ x < y \Rightarrow \pi_1(x) \leq \pi_1(y), \text{ with other words } \pi_1 \text{ is increasing} \\ \pi_3(0) = 1 \text{ and } \pi_3(1) = 0 \\ x < y \Rightarrow \pi_3(x) \geq \pi_3(y), \text{ with other words } \pi_3 \text{ is decreasing} \end{cases}$$

For the parameter values $(\alpha,\beta,\gamma,\delta) \in \mathbb{R}^4$ it must hold that:

$$\alpha \leq \beta \leq \gamma \leq \delta$$

For each pair of continuous mappings π_1 and π_3 which satisfy the border conditions and for each $(\alpha,\beta,\gamma,\delta) \in \mathbb{R}^4 : \alpha \leq \beta \leq \gamma \leq \delta$ the Π-function is formally defined by:

$$\Pi(.;\alpha,\beta,\gamma,\delta) : U \rightarrow [0,1]$$
$$x \mapsto \begin{cases} 1 & \text{iff } (x < \alpha) \wedge (\alpha = \beta = \inf(U)) \\ 0 & \text{iff } (x < \alpha) \wedge (\alpha < \beta) \\ \pi_1(\dfrac{x-\alpha}{\beta-\alpha}) & \text{iff } (\alpha \leq x < \beta) \wedge (\alpha < \beta) \\ 1 & \text{iff } \beta \leq x \leq \gamma \\ \pi_3(\dfrac{x-\gamma}{\delta-\gamma}) & \text{iff } (\gamma < x \leq \delta) \wedge (\gamma < \delta) \\ 0 & \text{iff } (x > \delta) \wedge (\gamma < \delta) \\ 1 & \text{iff } (x > \delta) \wedge (\gamma = \delta = \sup(U)) \end{cases}$$

Due to its generality, the Π-function can, with adequate parameter values and adequate choices for the functions π_1 and π_3, result in an S-function or in a π-function.

2.1.5.4 The trapezoidal functions

Trapezoidal functions are obtained by choosing the functions π_1 and π_3 for the construction of a Π-function as follows:

$$\begin{aligned} \pi_1 : [0,1] &\rightarrow [0,1] \\ x &\mapsto x \end{aligned} \qquad \text{and} \qquad \begin{aligned} \pi_3 : [0,1] &\rightarrow [0,1] \\ x &\mapsto 1 - x \end{aligned}$$

Example (c) in figure 2.4 illustrates a trapezoidal shape function. Trapezoidal shape functions are fully characterized by the four domain values α, β, γ and δ, which are defined as depicted in the figure. For defining a trapezoidal function it is sufficient to provide these four values.

Due to their simplicity, trapezoidal shape functions are very frequently used in 'fuzzy' databases and 'fuzzy' querying.

2.1.5.5 The triangular functions

A triangular shape function is a special case of a trapezoidal shape function where $\beta = \gamma$. Example (d) in figure 2.4 illustrates a triangular shape function. Although triangular shape functions are characterized by only three domain values α, β and δ, they are not frequently used in the context of 'fuzzy' databases due to their limited modelling abilities.

2.1.6 Interpretation of fuzzy sets

The membership grades $\mu_{\tilde{V}}(x)$ of the elements $x \in U$ of a fuzzy set $\tilde{V}$ can be *interpreted* in three different ways [5]: as degrees of compatibility, as degrees of truth, or as degrees of uncertainty.

2.1.6.1 Degree of compatibility

In his original, historical oldest, interpretation a membership grade $\mu_{\tilde{V}}(x)$ is considered to be a degree that expresses to which extent the element x is compatible with the 'prototype'-elements of $\tilde{V}$ (i.e., is compatible with the elements that are (or should be) full elements of $\tilde{V}$). Such a fuzzy set has intrinsically a conjunctive interpretation: the fuzzy set represents a collection of elements which together are representative for the concept that is being modelled.

Example 2.1. An example of a fuzzy set where the membership grades are interpreted as degrees of compatibility, is the fuzzy set of 'expensive prices' for paintings. The membership function of this fuzzy set could for example be modelled by means of shape function (a) of figure 2.4. From such a membership function we can for example derive that the membership grade of 100.000, and thus also the extent to which 100.000 is compatible with an expensive price, equals 0.42. $\diamond$

2.1.6.2 Degree of truth

With the naming 'degree of truth' we prefer to use the terminology of L.A. Zadeh [29]. D. Dubois & H. Prade call this kind of interpretation 'degree of preference' [5]. With this interpretation, the membership grade $\mu_{\tilde{V}}(x)$ expresses to which extent the element x of the fuzzy set $\tilde{V}$ is true (or applies). Also in this case the fuzzy set intrinsically has a conjunctive interpretation. With other words, all elements of the fuzzy set together represent the concept that is being modelled.

Example 2.2. An example of a fuzzy set where each membership grade represents a degree of truth is the fuzzy set that models the languages spoken by a person. This fuzzy set can for example be specified by:

$$\{(English, 1), (French, 0.6), (Spanish, 0.8), (German, 0.2)\}$$

which represents a 'perfect' knowledge of English (highest truth), a good knowledge of Spanish, a moderate knowledge of French and a limited knowledge of the German language. $\diamond$

2.1.6.3 Degree of uncertainty

This interpretation has been presented by L.A. Zadeh with the introduction of the *possibility theory* [26]. Hereby, the membership grade $\mu_{\tilde{V}}(x)$ represents the extent to which it is possible that a single-valued parameter p —of which it is known that the parameter takes values in the universe of discourse U— equals the value x. Thus the fuzzy set $\tilde{V}$ hereby represents all allowed values that are considered to be possible as value for p. Because p is single-valued —due to which p can take only one single value at the same time— and because the fuzzy set is a representation of the actual value of p, the interpretation of the fuzzy set $\tilde{V}$ is disjunctive.

Example 2.3. As example, we consider the price p of a painting. If p is not exactly known, but it is known that the painting is expensive, then the value of p can be modelled by a membership function like the one represented in figure 2.4 (a). The interpretation of the membership function is now different than in example 2.1: if a painting is for sale, we know that it has some price. The possible values of this price are determined by the fuzzy set. As such, it is for example uncertain, but possible to an extent 0.42 that the price of the painting is 100.000. ◇

Possibility theory will be described in the next section.

2.2 Possibility theory

Possibility theory [26, 4, 6] is closely related to fuzzy set theory. It is used for the modelling of uncertainty and is thus an alternative for probability theory. Possibility theory departs from a more conservative approach, due to which uncertainty is modelled with less risk for errors, but also with less information for the user. This makes the theory especially suited for the modelling of uncertainty in situations where the user has no full control (over the experiment), what is mostly the case with uncertainty modelling in the context of database applications. Possibility theory should not be seen as a competitor of **probability theory**. In contrary, in cases of full control (over the experiment) the use of probability theory [15, 8] is better because this provides the user with more information. Although possibility theory is nowadays almost always used for the modelling of uncertainty in the context of ‘fuzzy’ databases, it is surely not excluded that in the future both theories will be applied next to each other. In fact, the theory of imprecise probabilities [17, 18], which acts as a uniform framework for both probability and possibility theory, and the use of generalized constraints [28, 29], are first research steps in this direction. In the remainder of this section we describe the basic concepts of possibility theory in more detail.

2.2.1 Fuzzy measures

The concept of **fuzzy measure** is important for the modelling of the uncertainty that can exist about the fact whether a given element of a universe of discourse is element or is not element of a set of the set of all subsets of that universe. To model such uncertainty a membership grade ($\in [0,1]$), that is interpreted as a degree of uncertainty, is associated with each subset. This degree of uncertainty then denotes to which extent it is certain that the element belongs to that subset.

Two specific types of fuzzy measures play an important role in the possibility theory: the **possibility measures** and the **necessity measures**.

Definition 2.16 (Possibility measure) *With the understanding that U is a universe of discourse, a possibility measure over the power set $\wp(U)$ is defined by a function*

$$\mathrm{Pos} : \wp(U) \to [0,1]$$

which satisfies the following axioms:

- *Axiom p1.* $\mathrm{Pos}(\emptyset) = 0$ *and* $\mathrm{Pos}(U) = 1$ *(border conditions).*
- *Axiom p2.* $\forall\, A,B \in \wp(U) : A \subseteq B \Rightarrow \mathrm{Pos}(A) \leq \mathrm{Pos}(B)$
  (monotonicity).

and for which it holds for each set

$$\{A_k | k \in K\} \subseteq \wp(U)$$

with K an arbitrary index set, that

$$\mathrm{Pos}\left(\bigcup_{k\in K} A_k\right) = \sup_{k\in K} \mathrm{Pos}(A_k)$$

□

By definition each possibility measure has an associated necessity measure. The relationship between both of them is as follows:

With the understanding that U is a universe of discourse and Pos is a possibility measure that is defined over the power set $\wp(U)$, the necessity measure Nec that is associated with Pos is determined by

$$\begin{aligned} \mathrm{Nec} : \wp(U) &\to [0,1] \\ A &\mapsto \mathrm{Nec}(A) = 1 - \mathrm{Pos}(\overline{A}) \end{aligned}$$

A necessity measure can also be formally defined as follows:

Definition 2.17 (Necessity measure) *With the understanding that U is a universe of discourse, a necessity measure over the power set $\wp(U)$ is defined by a function*

$$\mathrm{Nec} : \wp(U) \rightarrow [0,1]$$

which satisfies the following axioms:

- *Axiom n1.* $\mathrm{Nec}(\emptyset) = 0$ *and* $\mathrm{Nec}(U) = 1$ *(border conditions).*
- *Axiom n2.* $\forall A, B \in \wp(U) : A \subseteq B \Rightarrow \mathrm{Nec}(A) \leq \mathrm{Nec}(B)$ *(monotonicity).*

and for which it holds for each set

$$\{A_k | k \in K\} \subseteq \wp(U)$$

with K an arbitrary index set, that

$$\mathrm{Nec}\left(\bigcap_{k\in K} A_k\right) = \inf_{k\in K} \mathrm{Nec}(A_k)$$

□

A possibility measure Pos which is defined over the power set $\wp(U)$ denotes the extent $\mathrm{Pos}(A)$ to which it is considered to be *possible* that an element of the universe belongs to the set $A \in \wp(U)$. A necessity measure Nec which is defined over the power set $\wp(U)$ denotes the extent $\mathrm{Nec}(A)$ to which it is considered to be *necessary* or *certain* that an element of the universe belongs to the set $A \in \wp(U)$.

From definitions 2.16 and 2.17 the following important properties can be derived:

- $\forall A, B \in \wp(U) : \mathrm{Pos}(A \cup B) = \max(\mathrm{Pos}(A), \mathrm{Pos}(B))$
- $\forall A, B \in \wp(U) : \mathrm{Nec}(A \cap B) = \min(\mathrm{Nec}(A), \mathrm{Nec}(B))$

Furthermore it can be proved that:

- $\forall A, B \in \wp(U) : \mathrm{Pos}(A \cap B) \leq \min(\mathrm{Pos}(A), \mathrm{Pos}(B))$
- $\forall A, B \in \wp(U) : \mathrm{Nec}(A \cup B) \geq \max(\mathrm{Nec}(A), \mathrm{Nec}(B))$

Only in the special case that A and B are independent, i.e. if A and B are sets that are determined by independent events, it holds that:

- $\forall A, B \in \wp(U) : \mathrm{Pos}(A \cap B) = \min(\mathrm{Pos}(A), \mathrm{Pos}(B))$
- $\forall A, B \in \wp(U) : \mathrm{Nec}(A \cup B) = \max(\mathrm{Nec}(A), \mathrm{Nec}(B))$

Because $A \cup \overline{A} = U$ we obtain furthermore that

$$\max(\mathrm{Pos}(A), \mathrm{Pos}(\overline{A})) = \mathrm{Pos}(A \cup \overline{A}) = \mathrm{Pos}(U) = 1$$

so that we can derive that

$$\mathrm{Pos}(A) < 1 \Rightarrow \mathrm{Pos}(\overline{A}) = 1$$

with other words, if A is not fully possible, then its complement $\overline{A}$ is fully possible, or also either A or its complement $\overline{A}$ is always fully possible. It also follows that

$$\mathrm{Pos}(A) + \mathrm{Pos}(\overline{A}) \geq 1.$$

Analoguously, based on $A \cap \overline{A} = \emptyset$, we can find that

$$\min(\mathrm{Nec}(A), \mathrm{Nec}(\overline{A})) = \mathrm{Nec}(A \cap \overline{A}) = \mathrm{Nec}(\emptyset) = 0$$

from which we can derive that

$$\mathrm{Nec}(A) > 0 \Rightarrow \mathrm{Nec}(\overline{A}) = 0.$$

It also follows that

$$\mathrm{Nec}(A) + \mathrm{Nec}(\overline{A}) \leq 1.$$

Based on the relationship between possibility measures and necessity measures it can be proved that:

$$(\mathrm{Nec}(A) > 0) \Rightarrow (\mathrm{Pos}(A) = 1)$$

and

$$(\mathrm{Pos}(A) < 1) \Rightarrow (\mathrm{Nec}(A) = 0).$$

Indeed:

- Assume that $\mathrm{Nec}(A) > 0$, then $\mathrm{Nec}(\overline{A}) = 0$, so that

$$\mathrm{Pos}(A) = 1 - \mathrm{Nec}(\overline{A}) = 1.$$

- Assume that $\mathrm{Pos}(A) < 1$, then $\mathrm{Pos}(\overline{A}) = 1$, so that

$$\mathrm{Nec}(A) = 1 - \mathrm{Pos}(\overline{A}) = 0.$$

2.2.2 Possibility distributions

Each possibility measure Pos that is defined over the power set $\wp(U)$ of a universe of discourse U, is uniquely determined by a so-called **possibility distribution**

$$\begin{aligned} \pi : U &\to [0,1] \\ x &\mapsto \mathrm{Pos}(\{x\}) \end{aligned}$$

If U is a universe with a finite number of elements, then it holds that

$$\forall\, A \in \wp(U) : \mathrm{Pos}(A) = \max_{x \in A} \pi(x)$$

For a universe U with an infinite number of elements this is generalized as

$$\forall\, A \in \wp(U) : \mathrm{Pos}(A) = \sup_{x \in A} \pi(x)$$

The possibility distribution π thus completely determines the possibility measure Pos and thus plays the same role as a probability distribution in probability theory.

From the relationship between possibility measure Pos and necessity measure Nec it follows that a possibility distribution π also completely determines a necessity measure Nec as follows: For a universe U with a finite number of elements, it holds that

$$\forall\, A \in \wp(U) : \text{Nec}(A) = \min_{x \notin A}(1 - \pi(x))$$

For a universe U with an infinite number of elements this is generalized as

$$\forall\, A \in \wp(U) : \text{Nec}(A) = \inf_{x \notin A}(1 - \pi(x)).$$

2.2.3 Possibility distributions and fuzzy sets

A possibility distribution can be derived form a **fuzzy set** of which the membership grades are interpreted as degrees of uncertainty.

Consider therefore the fuzzy set

$$\tilde{V} = \{(x, \mu_{\tilde{V}}(x)) | x \in U \wedge \mu_{\tilde{V}}(x) > 0\}$$

which is defined over a universe of discourse U and of which the membership grades are interpreted as degrees of uncertainty. Such a fuzzy set can be associated with a (single-valued) variable X, by which the fuzzy set puts a flexible constraint on the actual value of X. A membership grade $\mu_{\tilde{V}}(x)$, $x \in U$ hereby expresses to which extent it is possible that $X = x$. The possibility $\pi_X(x)$ that $X = x$ is with other words determined by the membership grade $\mu_{\tilde{V}}(x)$, i.e.

$$\forall\, x \in U : \pi_X(x) = \mu_{\tilde{V}}(x).$$

The function $\pi_X : U \to [0,1]$ that is defined by the previous consideration is clearly a possibility distribution over U, of which the associated possibility measure Pos_X is for each set $A \in \wp(U)$ defined by:

$$\text{Pos}_X(A) = \sup_{x \in A} \pi_X(x)$$

We say that the possibility distribution π_X follows the membership function $\mu_{\tilde{V}}$ [4].

If the fuzzy set $\tilde{V}$ is **normalized**, then the associated necessity measure Nec_X can for each set $A \in \wp(U)$ be derived from

$$\text{Nec}_X(A) = 1 - \text{Pos}_X(\overline{A}).$$

With this we are ready to clarify the more conservative nature of possibility theory (compared to probability theory). We have also introduced enough concepts to compare both theories to each other.

For a discrete probability distribution Pr the normalization condition requires that the sum of all occurring probabilities must be equal to one [8], i.e.

$$\sum_{x\in U} Pr(x) = 1.$$

For a continuous probability distribution this is generalized as

$$\int_U Pr(x)dx.$$

For possibility distributions the normalization condition is less stringent. Indeed the only requirement is that at least one possibility must be equal to one. For a discrete possibility distribution π this is expressed by

$$\max_{x\in U} \pi(x) = 1.$$

For a continuous possibility distribution this is generalized as

$$\sup_{x\in U} \pi(x) = 1.$$

Possibilities express preferences, which implies that a single possibility is in fact not informative, unless it is placed next to and compared with other possibilities. A possibility degree that equals 1 can correspond as well to an event that is completely certain (if the opposite event is completely impossible, i.e. has an associated possibility 0), as to an event that is not necessarily certain (if the opposite event is also completely possible). Probabilities however express information with respect to the relative occurrence of an event (the probability of an event A is the frequency with which A occurs). Due to the fact that the probability of the opposite event can be derived from the probability of the event, there is no need for a second measure in probability theory.

There exist several interrelationships between possibilities and probabilities. The relationships proposed by L.A. Zadeh stem from the following ideas:

1. What is not possible, is also not certain.
2. What is certain, is also possible.

With the notation $\text{Prob}(A)$ for the probability of an event A, the previous ideas can be translated as:

$$\forall\, A \in \wp(U) : \text{Prob}(A) \leq \text{Pos}(A)$$

and

$$\forall\, A \in \wp(U) : \text{Nec}(A) \leq \text{Prob}(A).$$

These and other relationships between both theories are described in more detail in [16, 10, 9].

2.3 Fuzzy logic

Fuzzy logic is an extension of Boolean logic where the classical truth values ‘true’ (T) and ‘false’ (F) are extended to fuzzy truth values which are elements of the unit interval $[0,1]$. Hereby, the classical truth value ‘true’ (T) corresponds to 1, the truth value ‘false’ (F) to 0. The intermediate values ($\in]0,1[$) denote a gradual truth: the closer to 1, the more the truth value is true; the closer to 0, the more the truth value is false. The semantics of a fuzzy truth value $x \in [0,1]$ is are illustrated in figure 2.5.

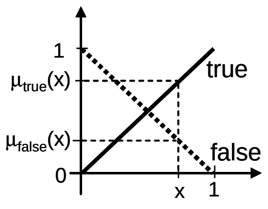

Fig. 2.5 Fuzzy truth values.

The truth values ‘true’ and ‘false’ can be seen as linguistic terms which are modelled by the visualized membership functions that are defined over the unit interval. The membership grade $\mu_{true}(x)$ denotes to which extent x is compatible with ‘true’ and analogously the membership grade $\mu_{false}(x)$ denotes to which extent x is compatible with ‘false’.

Taking into account the above mentioned relationship with fuzzy theory, the logical operators of conjunction ($\wedge$), disjunction ($\vee$), negation ($\neg$), implication ($\Rightarrow$) and equivalence ($\Leftrightarrow$) are defined conform to their algebraic counterparts in fuzzy set theory, i.e.

- **Conjunction**: $\forall\, x,y \in [0,1] : x \wedge y = i(x,y)$.
- **Disjunction**: $\forall\, x,y \in [0,1] : x \vee y = u(x,y)$.
- **Negation**: $\forall\, x \in [0,1] : \neg(x) = 1 - x$.
- **Implication**: $\forall\, x,y \in [0,1] : x \Rightarrow y = u(1-x,y)$.
- **Equivalence**: $\forall\, x,y \in [0,1] : x \Leftrightarrow y = i(u(1-x,y),u(1-y,x))$.

Which can, with the use of the Zadeh t-norm min and Zadeh t-conorm max, be concretised as:

- **Conjunction**: $\forall\, x,y \in [0,1] : x \wedge y = \min(x,y)$.
- **Disjunction**: $\forall\, x,y \in [0,1] : x \vee y = \max(x,y)$.
- **Negation**: $\forall\, x \in [0,1] : \neg(x) = 1 - x$.
- **Implication**: $\forall\, x,y \in [0,1] : x \Rightarrow y = \max(1-x,y)$.
- **Equivalence**:

$$\forall\, x,y \in [0,1] : x \Leftrightarrow y = \min(\max(1-x,y),\max(1-y,x)).$$

2.4 (Extended) Possibilistic truth values

In the context of ‘fuzzy’ databases uncertainty (about truth) is mostly modelled by means of possibility measures and necessity measures. This approach has some disadvantages when considering logical operations. As such, with respect to a possibility measure, it always holds for the union operator that

$$\text{Pos}(A \cup B) = \max(\text{Pos}(A), \text{Pos}(B))$$

whereas for the intersection operator the equation

$$\text{Pos}(A \cap B) = \min(\text{Pos}(A), \text{Pos}(B))$$

only applies if A and B are independent of each other. Analogously, with respect to a necessity measure, it always holds for the intersection operator that

$$\text{Nec}(A \cap B) = \min(\text{Nec}(A), \text{Nec}(B))$$

whereas for the union operator the equation

$$\text{Nec}(A \cup B) = \max(\text{Nec}(A), \text{Nec}(B))$$

only applies if A and B are independent of each other.

Another disadvantage is that possibility and necessity measures have some restrictions with respect to the modelling and handling of missing information, what often occurs in databases[1].

Among others because of the previous disadvantages the so-called **possibilistic truth values** and **extended possibilistic truth values** have been developed. These allow to model linguistic uncertainty about the truth value of a proposition more adequately. Linguistic uncertainty is the uncertainty that is contained in affirmative propositions like for example ‘the painting is cheap’ and ‘the water temperature is high’. The truth value of such a proposition gives information about for example the temperature of the water without having complete knowledge about the actual temperature of the water.

2.4.1 Extended truth values

By using classical two-valued Boolean propositional logic cases of doubt about the truth value of a proposition can not adequately be modelled. This is due to the fact that in two-valued logic it is explicitly assumed that each proposition can be evaluated and is either *completely true*, or *completely false*. This assumption is commonly known as the **principle of bivalence** and states that

[1] The handling of missing information in databases is described in chapter 4.

Each proposition is either true, or false.

Following this philosophy, a **truth value** can be associated with each proposition, denoting whether the proposition is true (T) or false (F). With the understanding that P represents the universe of all propositions and $I = \{T, F\}$ is a set with total ordering relation

$$\leq = \{(F,F),(F,T),(T,T)\},$$

a truth value can be formally defined by:

Definition 2.18 (Truth value) *The truth value $t(p)$ of a proposition $p \in P$ is formally defined by means of the function t:*

$$t : P \to I : p \mapsto t(p)$$

$t(p)$ equals T if p is true, i.e. if p corresponds to the reality. Else $t(p)$ equals F.
□

Example 2.4. • t('2 is an even number'$) = T$
• t('Eddy Merckx was a famous basketball player'$) = F$
◇

In database and information systems it occurs that data are missing[2] due to the fact that they relate to exceptions which have not been dealt with explicitly in the database design. Examples are the flying speed of penguins in a record of record type 'bird', the number of habitants of the Eiffel tower in a record of record type 'building' and the salary of a retired person in a record of record type 'person'. In such cases data are missing because they do not apply. Modelling these cases with a regular domain value will not allow us the track the cause of the missing and brings along with it some kind of information loss which in some cases can cause annoying side effects in query results. For example, when querying the database for 'birds that cannot fly fast' or 'buildings with few habitants' or 'persons with low salaries' we do not want penguins, monuments like the Eiffel tower and retired persons to be treated as regular query answers.

To avoid such information loss and side effects it is required to use a semantic richer logical framework which explicitly allows to reflect inapplicability. The principle of bivalence is then replaced by the **principle of trivalence** which states that

Each proposition either evaluates to true, false or inapplicable, where the latter indicates that the truth value of the proposition is not defined.

To formally support the principle of trivalence, the concept **extended truth value** has been introduced. Hereby, definition 2.18 is extended with an extra truth value $\perp$ which models 'undefined'. Considering the extended set $I^* = \{T, F, \perp\}$, an extended truth value can be formally defined by:

[2] The handling of missing information is more extensively described in chapter 4.

Definition 2.19 (Extended truth value) *The extended truth value $t^*(p)$ of a proposition $p \in P$ is formally defined by means of the function t^*:*

$$t^* : P \rightarrow I^* : p \mapsto t^*(p)$$

where

- *$t^*(p) = T$, if p corresponds with reality, i.e. if p is true;*
- *$t^*(p) = F$, if p does not correspond with reality, i.e. if p is false;*
- *$t^*(p) = \perp$, if proposition p is (partially) not applicable, undefined, not existent, or not supported; in such cases it does not make sense to decide whether p corresponds with reality or not, i.e. p is then neither true, nor false, but undefined.*

$\square$

Logical operators can be used to build a new proposition that is composed of other propositions. The following logical operators are provided:

- Negation: $NOT : P \rightarrow P : p \mapsto NOT(p)$
- Conjunction: $AND : P \times P \rightarrow P : (p,q) \mapsto p\ AND\ q$
- Disjunction: $OR : P \times P \rightarrow P : (p,q) \mapsto p\ OR\ q$
- Implication: $IF-THEN : P \times P \rightarrow P : (p,q) \mapsto IF\ p\ THEN\ q$
- Equivalence: $IFF : P \times P \rightarrow P : (p,q) \mapsto p\ IFF\ q$

A proposition without logical operators is called a *simple proposition*. Examples are ‘2 is an even number’, ‘Lance Armstrong is a famous basketball player’, ‘x is larger than 1.000’, etc. A proposition that is obtained as a combination of other (simple) propositions and logical operators is called a *composite proposition*. Examples are ‘2 is an even number AND 2 is prime’, ‘Lance Armstrong is a famous basketball player AND Lance Armstrong is an American’, ‘x is larger than 1.000 OR x is smaller than 10’.

The *extended truth value* of a composite proposition can be computed by means of the following computation rules:

- *Rule for negation:*

$$\forall\ p \in P : t^*(NOT\ p) = \neg(t^*(p))$$

  where $\neg : I^* \rightarrow I^* : x \mapsto \neg(x)$ is defined by the truth table

x	$\neg(x)$
T	F
F	T
$\perp$	$\perp$

- *Rule for conjunction:*

$$\forall\ p,q \in P : t^*(p\ AND\ q) = t^*(p) \wedge t^*(q)$$

  where $\wedge : I^* \times I^* \rightarrow I^* : (x,y) \mapsto x \wedge y$ is defined by the truth table

x	y	$x \wedge y$
T	T	T
T	F	F
T	$\perp$	$\perp$
F	T	F
F	F	F
F	$\perp$	F
$\perp$	T	$\perp$
$\perp$	F	F
$\perp$	$\perp$	$\perp$

- *Rule for disjunction:*

$$\forall\, p,q \in P : t^*(p\ OR\ q) = t^*(p) \vee t^*(q)$$

where $\vee : I^* \times I^* \to I^* : (x,y) \mapsto x \vee y$ is defined by the truth table

x	y	$x \vee y$
T	T	T
T	F	T
T	$\perp$	T
F	T	T
F	F	F
F	$\perp$	$\perp$
$\perp$	T	T
$\perp$	F	$\perp$
$\perp$	$\perp$	$\perp$

- *Rule for implication:*

$$\forall\, p,q \in P : t^*(IF\ p\ THEN\ q) = t^*(p) \Rightarrow t^*(q)$$

where $\Rightarrow : I^* \times I^* \to I^* : (x,y) \mapsto x \Rightarrow y$ is defined by the truth table

x	y	$x \Rightarrow y$
T	T	T
T	F	F
T	$\perp$	$\perp$
F	T	T
F	F	T
F	$\perp$	T
$\perp$	T	T
$\perp$	F	$\perp$
$\perp$	$\perp$	p$\perp$

Remark that: $t^*(IF\ p\ THEN\ q) = \neg(t^*(p)) \vee t^*(q)$
$\qquad = t^*(NOT\ p\ OR\ q)$

- *Rule for equivalence:*

$$\forall\, p,q \in P : t^*(p\ \mathit{IFF}\ q) = t^*(p) \Leftrightarrow t^*(q)$$

where $\Leftrightarrow: I^* \times I^* \to I^* : (x,y) \mapsto x \Leftrightarrow y$ is defined by the truth table

x	y	$x \Leftrightarrow y$
T	T	T
T	F	F
T	$\perp$	$\perp$
F	T	F
F	F	T
F	$\perp$	$\perp$
$\perp$	T	$\perp$
$\perp$	F	$\perp$
$\perp$	$\perp$	$\perp$

Remark that:

$$\begin{aligned} t^*(p\ \mathit{IFF}\ q) &= (t^*(p) \Rightarrow t^*(q)) \wedge (t^*(q) \Rightarrow t^*(p)) \\ &= t^*((\mathit{IF}\ p\ \mathit{THEN}\ q)\ \mathit{AND}\ (\mathit{IF}\ q\ \mathit{THEN}\ p)) \end{aligned}$$

The operator $\neg$ has precedence over the operators $\wedge$ and $\vee$ and the operator $\wedge$ has precedence over the operator $\vee$. The evaluation of an expression is from left to right, unless brackets are used to enforce precedence.

Due to the previous calculation rules the resulting logical framework is a strong three-valued Kleene logic [14]. Kleene logics are truth functional, which means that according to these logics the behaviour of each logical operator is reflected by a logical function that combines Kleene truth values [1]. Thanks to this property the *extended truth value* of each composite proposition can be computed as a function of the extended truth values of its original propositions.

The natural partial ordering relation relation $\leq$ over I^* that corresponds with the algebraic structure $(I^*, \wedge, \vee)$ satisfies:

$$\leq = \{(F,F),(F,\perp),(F,T),(\perp,\perp),(\perp,T),(T,T)\}$$

2.4.2 Extended possibilistic truth values

In reality, a lot of situations exist where we can not unambiguously determine whether the truth value of a proposition is either *completely* true, *completely* false or *completely* undefined. Examples of such propositions are e.g., ‘the house is cheap’, ‘the car is fast’ and ‘the suspect is tall’. The extended truth values that have been introduced in the previous section 2.4.1 —only— allow it to represent the actual truth value of a proposition that is a priori considered to be either completely true, false or undefined. What is required, is a more epistemological representation of the truth of a proposition which allows it to better reflect our knowledge about the actual truth value.

In his approach to obtain such a representation, Prade [13] uses a possibility measure to generalize classical two-valued truth values. Each proposition then evaluates to a so-called possibilistic truth value which is defined over the universe $\{T,F\}$. This approach has been further developed in [1]. A basic overview of this is given in the following subsection. Next, the concept extended possibilistic truth value is described.

2.4.2.1 Possibilistic truth values

Definition.

By generalizing the classical truth values true (T) and false (F), the principle of bivalence is replaced by a more general **principle of valence**, which states that:

Each proposition has a (possibilistic) truth value.

For this generalization, the set $\tilde{\wp}(I)$ of all fuzzy sets that are defined over the universe $I = \{T,F\}$ is considered.

Definition 2.20 (Possibilistic truth value) *The possibilistic truth value $\tilde{t}(p)$ of a proposition $p \in P$ is formally defined by means of the function $\tilde{t}$:*

$$\tilde{t} : P \to \tilde{\wp}(I) : p \mapsto \tilde{t}(p)$$

which associates a fuzzy set $\tilde{t}(p)$ with each $p \in P$. The fuzzy set $\tilde{t}(p)$ represents a possibility distribution, i.e. its membership grades are interpreted as degrees of uncertainty:

$$\forall\, x \in I : \pi_{t(p)}(x) = \mu_{\tilde{t}(p)}(x)$$

i.e.

$$\forall\, p \in P : \pi_{t(p)} = \tilde{t}(p)$$

□

A **possibilistic truth value** is thus a fuzzy set, that generally has the following form

$$\tilde{t}(p) = \{(T, \mu_{\tilde{t}(p)}(T)), (F, \mu_{\tilde{t}(p)}(F))\}.$$

Interpretation.

By definition 2.20 the possibilistic truth value $\tilde{t}(p)$ of a proposition $p \in P$ must be interpreted as follows:

$$\mathrm{Pos}(t(p) = \{T\}) = \mu_{\tilde{t}(p)}(T)$$

and

$$\mathrm{Pos}(t(p) = \{F\}) = \mu_{\tilde{t}(p)}(F)$$

where Pos represents a possibility measure.
Special cases of possibilistic truth values are:

Truth value $\tilde{t}(p)$	Interpretation
$\{(T,1)\}$	p is true
$\{(F,1)\}$	p is false
$\{(T,1),(F,1)\}$	p is unknown

By using Zadeh's standard equality definition for fuzzy sets (cf. definition 2.11), the equality operator for possibilistic truth values can be defined by:

Definition 2.21 (Equality) $\forall\, p,p' \in P$:

- $\tilde{t}(p) = \tilde{t}(p') \Leftrightarrow \forall\, x \in I : \mu_{\tilde{t}(p)}(x) = \mu_{\tilde{t}(p')}(x)$
- $\tilde{t}(p) \neq \tilde{t}(p') \Leftrightarrow \exists\, x \in I : \mu_{\tilde{t}(p)}(x) \neq \mu_{\tilde{t}(p')}(x)$

□

Example 2.5. The possibilistic truth value

$$\tilde{t}(\text{'the house is cheap'}) = \{(T,1.0),(F,0.7)\}$$

of the proposition 'the house is cheap' has to be interpreted as

$$\pi_{t(\text{'the house is cheap'})} = \{(T,1.0),(F,0.7)\}$$

i.e.

$$\text{Pos}(t(\text{'the house is cheap'}) = \{T\}) = 1.0$$

$$\text{Pos}(t(\text{'the house is cheap'}) = \{F\}) = 0.7$$

which states that it is completely possible (to an extent 1) that the house is cheap, but on the other hand it is also less possible (to an extent 0.7) that the house is not cheap. ◇

Calculus.

The computation rules for negation, conjunction, disjunction, implication and equivalence are defined as generalizations of their counterparts in classical two-valued logic. For the generalization, Zadeh's extension principle (cf. definition 2.15) can be used:

- *Rule for negation:*
$$\forall\, p \in P : \tilde{t}(NOT\ p) = \tilde{\neg}(\tilde{t}(p))$$
where $\tilde{\neg} : \tilde{\wp}(I) \to \tilde{\wp}(I) : \tilde{V} \mapsto \tilde{\neg}(\tilde{V})$ is obtained by applying Zadeh's extension principle to the operator $\neg$:
$$\mu_{\tilde{\neg}(\tilde{V})}(T) = \sup_{x \in \{x | x \in I \wedge \neg(x) = T\}} \mu_{\tilde{V}}(x) = \mu_{\tilde{V}}(F)$$

and

$$\mu_{\tilde{\neg}(\tilde{V})}(F) = \sup_{x \in \{x | x \in I \wedge \neg(x) = F\}} \mu_{\tilde{V}}(x) = \mu_{\tilde{V}}(T)$$

- *Rule for conjunction:*

$$\forall\, p,q \in P : \tilde{t}(p\ AND\ q) = \tilde{t}(p) \tilde{\wedge} \tilde{t}(q)$$

where $\tilde{\wedge} : \tilde{\wp}(I) \times \tilde{\wp}(I) \to \tilde{\wp}(I) : (\tilde{U},\tilde{V}) \mapsto \tilde{U}\ \tilde{\wedge}\ \tilde{V}$ is obtained by applying Zadeh's extension principle to the operator $\wedge$:

$$\begin{aligned}\mu_{\tilde{U}\tilde{\wedge}\tilde{V}}(T) &= \sup_{(x,y) \in \{(x,y)|(x,y) \in I \times I \wedge (x \wedge y = T)\}} \min(\mu_{\tilde{U}}(x), \mu_{\tilde{V}}(y)) \\ &= \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(T))\end{aligned}$$

and

$$\begin{aligned}\mu_{\tilde{U}\tilde{\wedge}\tilde{V}}(F) &= \sup_{(x,y) \in \{(x,y)|(x,y) \in I \times I \wedge (x \wedge y = F)\}} \min(\mu_{\tilde{U}}(x), \mu_{\tilde{V}}(y)) \\ &= \max\begin{pmatrix} \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(F)), \\ \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(T)), \\ \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(F)) \end{pmatrix}\end{aligned}$$

- *Rule for disjunction:*

$$\forall\, p,q \in P : \tilde{t}(p\ OR\ q) = \tilde{t}(p) \tilde{\vee} \tilde{t}(q)$$

where $\tilde{\vee} : \tilde{\wp}(I) \times \tilde{\wp}(I) \to \tilde{\wp}(I) : (\tilde{U},\tilde{V}) \mapsto tildeU\ \tilde{\vee}\ \tilde{V}$ is obtained by applying Zadeh's extension principle to the operator $\vee$:

$$\begin{aligned}\mu_{\tilde{U}\tilde{\vee}\tilde{V}}(T) &= \sup_{(x,y) \in \{(x,y)|(x,y) \in I \times I \wedge (x \vee y = T)\}} \min(\mu_{\tilde{U}}(x), \mu_{\tilde{V}}(y)) \\ &= \max\begin{pmatrix} \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(F)), \\ \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(T)), \\ \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(T)) \end{pmatrix}\end{aligned}$$

and

$$\begin{aligned}\mu_{\tilde{U}\tilde{\vee}\tilde{V}}(F) &= \sup_{(x,y) \in \{(x,y)|(x,y) \in I \times I \wedge (x \vee y = F)\}} \min(\mu_{\tilde{U}}(x), \mu_{\tilde{V}}(y)) \\ &= \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(F))\end{aligned}$$

- *Rule for implication:*

$$\forall\, p,q \in P : \tilde{t}(IF\ p\ THEN\ q) = \tilde{t}(p) \tilde{\Rightarrow} \tilde{t}(q)$$

where $\tilde{\Rightarrow} : \tilde{\wp}(I) \times \tilde{\wp}(I) \to \tilde{\wp}(I) : (\tilde{U},\tilde{V}) \mapsto \tilde{U}\ \tilde{\Rightarrow}\ \tilde{V}$ is obtained by applying Zadeh's extension principle to the operator $\Rightarrow$:

$$\mu_{\tilde{U}\tilde{\Rightarrow}\tilde{V}}(T) = \sup_{(x,y)\in\{(x,y)|(x,y)\in I\times I\wedge((x\Rightarrow y)=T)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y))$$
$$= \max\begin{pmatrix}\min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(F))\end{pmatrix}$$

and

$$\mu_{\tilde{U}\tilde{\Rightarrow}\tilde{V}}(F) = \sup_{(x,y)\in\{(x,y)|(x,y)\in I\times I\wedge((x\Rightarrow y)=F)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y))$$
$$= \min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(F))$$

Remark that: $\tilde{t}(IF\ p\ THEN\ q) = \tilde{t}(NOT\ p\ OR\ q)$

- *Rule for equivalence:*

$$\forall\, p,q \in P : \tilde{t}(p\ IFF\ q) = \tilde{t}(p)\tilde{\Leftrightarrow}\tilde{t}(q)$$

where $\tilde{\Leftrightarrow} : \tilde{\wp}(I)\times\tilde{\wp}(I)\to\tilde{\wp}(I) : (\tilde{U},\tilde{V})\mapsto \tilde{U}\tilde{\Leftrightarrow}\tilde{V}$ is obtained by applying Zadeh's extension principle to the operator $\Leftrightarrow$:

$$\mu_{\tilde{U}\tilde{\Leftrightarrow}\tilde{V}}(T) = \sup_{(x,y)\in\{(x,y)|(x,y)\in I\times I\wedge((x\Leftrightarrow y)=T)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y))$$
$$= \max\begin{pmatrix}\min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(F))\end{pmatrix}$$

and

$$\mu_{\tilde{U}\tilde{\Leftrightarrow}\tilde{V}}(F) = \sup_{(x,y)\in\{(x,y)|(x,y)\in I\times I\wedge((x\Leftrightarrow y)=F)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y))$$
$$= \max\begin{pmatrix}\min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(F)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(T))\end{pmatrix}$$

Remark that: $\tilde{t}(p\ IFF\ q) = \tilde{t}((IF\ p\ THEN\ q)\ AND\ (IF\ q\ THEN\ p))$

Partial ordering.

The natural partial ordering relation $\tilde{\leq}$ over $\tilde{\wp}(I)$ that corresponds to the algebraic structure $(\tilde{\wp}(I),\tilde{\wedge},\tilde{\vee})$ is obtained by applying the following proposition [1]:

Proposition 1 *The partial ordering relation $\tilde{\leq}$ over the set $\tilde{\wp}(I)$ satisfies:*

$$\forall\, \tilde{U},\tilde{V}\in\tilde{\wp}(I)\times\tilde{\wp}(I) : \tilde{U}\tilde{\leq}\tilde{V} \Leftrightarrow \begin{cases}\mu_{\tilde{U}}(T)\leq\mu_{\tilde{V}}(T)\\ \mu_{\tilde{U}}(F)\geq\mu_{\tilde{V}}(F)\end{cases}$$

∘

The proof of this proposition can be found in [1].

2.4.2.2 Extended possibilistic truth values

Definition.

In order to be able to efficiently deal with propositions of which the truth value is (partially) undefined, the concept 'extended truth value' has been generalized based on possibility theory [2]. The resulting truth values are called extended possibilistic truth values (EPTVs) and allow for a more flexible representation of the knowledge about the actual truth of a proposition. For the generalization the set $\tilde{\wp}(I^*)$ of all fuzzy sets that are defined over the universe $I^* = \{T, F, \perp\}$ is considered.

Definition 2.22 (Extended possibilistic truth value) *The extended possibilistic truth value $\tilde{t}^*(p)$ of a proposition $p \in P$ is formally defined by means of the function $\tilde{t}^*$:*

$$\tilde{t}^* : P \to \tilde{\wp}(I^*) : p \mapsto \tilde{t}^*(p)$$

which associates a fuzzy set $\tilde{t}^*(p)$ with each $p \in P$. The fuzzy set $\tilde{t}(p)$ represents a possibility distribution, i.e. its membership grades are interpreted as degrees of uncertainty:

$$\forall\, x \in I^* : \pi_{t^*(p)}(x) = \mu_{\tilde{t}^*(p)}(x)$$

i.e.

$$\forall\, p \in P : \pi_{t^*(p)} = \tilde{t}^*(p)$$

□

An **extended possibilistic truth value** is thus a fuzzy set, that generally has the following form

$$\tilde{t}^*(p) = \{(T, \mu_{\tilde{t}^*(p)}(T)), (F, \mu_{\tilde{t}^*(p)}(F)), (\perp, \mu_{\tilde{t}^*(p)}(\perp))\}.$$

Interpretation.

By definition 2.22 the extended possibilistic truth value $\tilde{t}^*(p)$ of a proposition $p \in P$ must be interpreted as follows:

$$\mathrm{Pos}(t^*(p) = \{T\}) = \mu_{\tilde{t}^*(p)}(T)$$
$$\mathrm{Pos}(t^*(p) = \{F\}) = \mu_{\tilde{t}^*(p)}(F)$$

and

$$\mathrm{Pos}(t^*(p) = \{\perp\}) = \mu_{\tilde{t}^*(p)}(\perp)$$

where Pos represents a possibility measure.
Special cases of extended possibilistic truth values are:

Truth value $\tilde{t}^*(p)$	Interpretation
$\{(T,1)\}$	p is true
$\{(F,1)\}$	p is false
$\{(T,1),(F,1)\}$	p is unknown
$\{(\perp,1)\}$	p is undefined
$\{(T,1),(F,1),(\perp,1)\}$	p is unknown or undefined

By using Zadeh's standard equality definition for fuzzy sets (cf. definition 2.11), the equality operator for extended possibilistic truth values can be defined by:

Definition 2.23 (Equality) $\forall\, p, p' \in P$:

- $\tilde{t}^*(p) = \tilde{t}^*(p') \Leftrightarrow \forall\, x \in I^* : \mu_{\tilde{t}^*(p)}(x) = \mu_{\tilde{t}^*(p')}(x)$
- $\tilde{t}^*(p) \neq \tilde{t}^*(p') \Leftrightarrow \exists\, x \in I^* : \mu_{\tilde{t}^*(p)}(x) \neq \mu_{\tilde{t}^*(p')}(x)$

□

Example 2.6. The extended possibilistic truth value

$$\tilde{t}^*(\text{‘the house is cheap’}) = \{(T,1.0),(F,0.7),(\perp,0.5)\}$$

of the proposition ‘the house is cheap’ is interpreted as

$$\pi_{t^*(\text{‘the house is cheap’})} = \{(T,1.0),(F,0.7),(\perp,0.5)\}$$

i.e.

$$\text{Pos}(t^*(\text{‘the house is cheap’}) = \{T\}) = 1.0$$

$$\text{Pos}(t^*(\text{‘the house is cheap’}) = \{F\}) = 0.7$$

and

$$\text{Pos}(t^*(\text{‘the house is cheap’}) = \{\perp\}) = 0.5$$

which means that it is completely possible (to an extent 1) that the house is cheap, it is less possible (to an extent 0.7) that the house is not cheap and on the other hand it is also possible to even lesser extent 0.5 that it does not make sense to consider prices for the house (e.g. because the house is not for sale).
◇

Calculus.

The computation rules for negation, conjunction, disjunction, implication and equivalence are defined as generalizations of their counterparts for extended truth values (which are given in section 2.4.1). A possible approach is to use Zadeh's extension principle (cf. definition 2.15) for this purpose:

- *Rule for negation:*

$$\forall\, p \in P : \tilde{t}^*(NOT\ p) = \tilde{\neg}(\tilde{t}^*(p))$$

where $\tilde{\neg} : \tilde{\wp}(I^*) \to \tilde{\wp}(I^*) : \tilde{V} \mapsto \tilde{\neg}(\tilde{V})$ is obtained by applying Zadeh's extension principle to the $\neg$:

$$\begin{aligned}
\mu_{\tilde{\neg}(\tilde{V})}(T) &= \sup_{x \in \{x | x \in I^* \wedge \neg(x) = T\}} \mu_{\tilde{V}}(x) = \mu_{\tilde{V}}(F)\\
\mu_{\tilde{\neg}(\tilde{V})}(F) &= \sup_{x \in \{x | x \in I^* \wedge \neg(x) = F\}} \mu_{\tilde{V}}(x) = \mu_{\tilde{V}}(T)\\
\mu_{\tilde{\neg}(\tilde{V})}(\perp) &= \sup_{x \in \{x | x \in I^* \wedge \neg(x) = \perp\}} \mu_{\tilde{V}}(x) = \mu_{\tilde{V}}(\perp)
\end{aligned}$$

- *Rule for conjunction:*

$$\forall\, p,q \in P : \tilde{t}^*(p\ AND\ q) = \tilde{t}^*(p) \tilde{\wedge} \tilde{t}^*(q)$$

where $\tilde{\wedge} : \tilde{\wp}(I^*) \times \tilde{\wp}(I^*) \to \tilde{\wp}(I^*) : (\tilde{U}, \tilde{V}) \mapsto \tilde{U}\ \tilde{\wedge}\ \tilde{V}$ is obtained by applying Zadeh's extension principle to the operator $\wedge$:

$$\begin{aligned}
\mu_{\tilde{U} \tilde{\wedge} \tilde{V}}(T) &= \sup_{(x,y) \in \{(x,y) | (x,y) \in I^* \times I^* \wedge (x \wedge y = T)\}} \min(\mu_{\tilde{U}}(x), \mu_{\tilde{V}}(y))\\
&= \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(T))
\end{aligned}$$

$$\begin{aligned}
\mu_{\tilde{U} \tilde{\wedge} \tilde{V}}(F) &= \sup_{(x,y) \in \{(x,y) | (x,y) \in I^* \times I^* \wedge (x \wedge y = F)\}} \min(\mu_{\tilde{U}}(x), \mu_{\tilde{V}}(y))\\
&= \max \begin{pmatrix} \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(F)),\\ \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(F)),\\ \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(\perp)),\\ \min(\mu_{\tilde{U}}(\perp), \mu_{\tilde{V}}(F)) \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
\mu_{\tilde{U} \tilde{\wedge} \tilde{V}}(\perp) &= \sup_{(x,y) \in \{(x,y) | (x,y) \in I^* \times I^* \wedge (x \wedge y = \perp)\}} \min(\mu_{\tilde{U}}(x), \mu_{\tilde{V}}(y))\\
&= \max \begin{pmatrix} \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(\perp)),\\ \min(\mu_{\tilde{U}}(\perp), \mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(\perp), \mu_{\tilde{V}}(\perp)) \end{pmatrix}
\end{aligned}$$

- *Rule for disjunction:*

$$\forall\, p,q \in P : \tilde{t}^*(p\ OR\ q) = \tilde{t}^*(p) \tilde{\vee} \tilde{t}^*(q)$$

where $\tilde{\vee} : \tilde{\wp}(I^*) \times \tilde{\wp}(I^*) \to \tilde{\wp}(I^*) : (\tilde{U}, \tilde{V}) \mapsto \tilde{U}\ \tilde{\vee}\ \tilde{V}$ is obtained by applying Zadeh's principle to the operator $\vee$:

$$\begin{aligned}\mu_{\tilde{U}\tilde{\vee}\tilde{V}}(T) &= \sup_{(x,y)\in\{(x,y)|(x,y)\in I^*\times I^*\wedge(x\vee y=T)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y)) \\ &= \max\begin{pmatrix}\min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(F)),\\ \min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(\perp)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(\perp),\mu_{\tilde{V}}(T))\end{pmatrix}\end{aligned}$$

$$\begin{aligned}\mu_{\tilde{U}\tilde{\vee}\tilde{V}}(F) &= \sup_{(x,y)\in\{(x,y)|(x,y)\in I^*\times I^*\wedge(x\vee y=F)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y)) \\ &= \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(F))\end{aligned}$$

$$\begin{aligned}\mu_{\tilde{U}\tilde{\vee}\tilde{V}}(\perp) &= \sup_{(x,y)\in\{(x,y)|(x,y)\in I^*\times I^*\wedge(x\vee y=\perp)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y)) \\ &= \max\begin{pmatrix}\min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(\perp)),\\ \min(\mu_{\tilde{U}}(\perp),\mu_{\tilde{V}}(F)),\\ \min(\mu_{\tilde{U}}(\perp),\mu_{\tilde{V}}(\perp))\end{pmatrix}\end{aligned}$$

- *Rule for implication:*

$$\forall\, p,q \in P : \tilde{t}^*(IF\ p\ THEN\ q) = \tilde{t}^*(p)\tilde{\Rightarrow}\tilde{t}^*(q)$$

  where $\tilde{\Rightarrow} : \tilde{\wp}(I^*)\times\tilde{\wp}(I^*)\rightarrow\tilde{\wp}(I^*) : (\tilde{U},\tilde{V})\mapsto\tilde{U}\,\tilde{\Rightarrow}\,\tilde{V}$ is obtained by applying Zadeh's extension principle to the operator $\Rightarrow$:

$$\begin{aligned}\mu_{\tilde{U}\tilde{\Rightarrow}\tilde{V}}(T) &= \sup_{(x,y)\in\{(x,y)|(x,y)\in I^*\times I^*\wedge((x\Rightarrow y)=T)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y)) \\ &= \max\begin{pmatrix}\min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(F)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(\perp)),\\ \min(\mu_{\tilde{U}}(\perp),\mu_{\tilde{V}}(T))\end{pmatrix}\end{aligned}$$

$$\begin{aligned}\mu_{\tilde{U}\tilde{\Rightarrow}\tilde{V}}(F) &= \sup_{(x,y)\in\{(x,y)|(x,y)\in I^*\times I^*\wedge((x\Rightarrow y)=F)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y)) \\ &= \min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(F))\end{aligned}$$

$$\begin{aligned}\mu_{\tilde{U}\tilde{\Rightarrow}\tilde{V}}(\perp) &= \sup_{(x,y)\in\{(x,y)|(x,y)\in I^*\times I^*\wedge((x\Rightarrow y)=\perp)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y)) \\ &= \max\begin{pmatrix}\min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(\perp)),\\ \min(\mu_{\tilde{U}}(\perp),\mu_{\tilde{V}}(F)),\\ \min(\mu_{\tilde{U}}(\perp),\mu_{\tilde{V}}(\perp))\end{pmatrix}\end{aligned}$$

Remark that: $\tilde{t}^*(IF\ p\ THEN\ q) = \tilde{t}^*(NOT\ p\ OR\ q)$

- *Rule for equivalence:*

$$\forall\, p,q \in P : \tilde{t}^*(p\ IFF\ q) = \tilde{t}^*(p) \tilde{\Leftrightarrow} \tilde{t}^*(q)$$

where $\tilde{\Leftrightarrow} : \tilde{\wp}(I^*) \times \tilde{\wp}(I^*) \to \tilde{\wp}(I^*) : (\tilde{U},\tilde{V}) \mapsto \tilde{U} \tilde{\Leftrightarrow} \tilde{V}$ is obtained by applying Zadeh's extension principle to the operator $\Leftrightarrow$:

$$\begin{aligned}\mu_{\tilde{U}\tilde{\Leftrightarrow}\tilde{V}}(T) &= \sup_{(x,y)\in\{(x,y)|(x,y)\in I^*\times I^*\wedge((x\Leftrightarrow y)=T)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y))\\ &= \max\begin{pmatrix}\min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(F))\end{pmatrix}\end{aligned}$$

$$\begin{aligned}\mu_{\tilde{U}\tilde{\Leftrightarrow}\tilde{V}}(F) &= \sup_{(x,y)\in\{(x,y)|(x,y)\in I^*\times I^*\wedge((x\Leftrightarrow y)=F)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y))\\ &= \max\begin{pmatrix}\min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(F)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(T))\end{pmatrix}\end{aligned}$$

$$\begin{aligned}\mu_{\tilde{U}\tilde{\Leftrightarrow}\tilde{V}}(\perp) &= \sup_{(x,y)\in\{(x,y)|(x,y)\in I^*\times I^*\wedge((x\Leftrightarrow y)=\perp)\}} \min(\mu_{\tilde{U}}(x),\mu_{\tilde{V}}(y))\\ &= \max\begin{pmatrix}\min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(\perp)),\\ \min(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(\perp)),\\ \min(\mu_{\tilde{U}}(\perp),\mu_{\tilde{V}}(T)),\\ \min(\mu_{\tilde{U}}(\perp),\mu_{\tilde{V}}(F)),\\ \min(\mu_{\tilde{U}}(\perp),\mu_{\tilde{V}}(\perp))\end{pmatrix}\end{aligned}$$

Remark that:

$$\tilde{t}^*(p\,IFF\ q) = \tilde{t}^*((IF\ p\,THEN\,q)\,AND\,(IF\,q\,THEN\ p))$$

2.5 Further developments

With the mathematical frameworks presented in this chapter, a new trend in information processing is set. Fuzzy set theory and possibility are the onset for the development of more advanced and more general frameworks and technologies like for example the framework for 'computing with words' [28, 30] and the theory of perceptions [31, 32]. These new developments create on their turn new opportunities for more advanced developments within the domain of 'fuzzy' databases. In this section we briefly describe what is meant by 'computing with words' and by 'fuzzy information granulation'.

'Computing with words' [28, 30] refers to a recent trend in information processing where one no longer uses numerical values in computations, but words that 'describe' numerical values in a way that is closer to natural language. Usually, these words correspond to the result of one or more (human) perceptions. The inspiration of this stems from the remarkable human capacity to execute complex physical or mental tasks (like e.g. driving a car in a garage box, playing golf, or summarizing a story) without a need for making numerical computations, but by just relying on (the processing of) perceptions (like e.g. speed, time, direction, or colour). Contrary to numerical values (which are precisely described or calculated), perceptions are often imprecisely or vaguely described by means of **words**.

The basic idea behind 'computing with words' is that (the semantics of) such words can be mathematically modelled by means of fuzzy set theory. Reasoning can then for example be done by using deduction rules that act on such words —fuzzy sets— and result in (new) words —fuzzy sets—, which on their turn can be used in deduction rules. 'Computing with words' can especially be used in situations where:

- The values are not adequately known, to justify a numerical representation.
- Precision is not required.
- The problem can not be solved, or the task can not be performed with numerical values.
- The concept is too complex for a numerical approach.

The 'theory of fuzzy information granulation' [29] forms the formal basis for 'computing with words'. Central in this theory are the granulation of information, the organization of information and the causual nature of information. Granulation relates to the decomposition of a whole into different parts; organization deals with the integration of parts in a whole; while the causual nature relates to the association of causes and consequences which are inherent to information.

An important concept in the 'theory of fuzzy information granulation', which is also important in the context of fuzzy databases, is the concept **generalized constraint** that is defined as follows:

Definition 2.24 (Generalized constraint) *With the understanding that X is a variable that is defined over universe of discourse U (i.e. X can only contain a value of U), the syntax of a generalized constraint is given as*

$$X \textit{ isr } R$$

Hereby, R is the constraining relationship and isr is a variable copula with a discrete variable r whose value defines the way how X is constrained by R. The semantics of a generalized constraint are completely determined by the value of r. □

The most important kinds of generalized constraints and their corresponding value for r are:

1. The **equality constraints**, $r = e$. The semantics of an equality constraint

$$X \textit{ ise } a$$

is defined by $X = a$.

2. The **possibilistic constraints**, $r = blanco$. In this case, X is interpreted as a disjunctive[3] variable —i.e. a variable which at the same time can contain only one value of the universe of discourse U—. Furthermore,

$$X \ is \ R$$

with R being a fuzzy set with membership function $\mu_R : U \rightarrow [0,1]$, means that a possibility distribution π_R is derived from R. This possibility distribution is considered to be the possibility distribution of X and expresses that the possibility that $X = x$, $x \in U$ is determined by

$$\pi_R(x) = \mu_R(x).$$

An example of a possibilistic constraint is

$$X \ is \ expensive \text{ which is equivalent with } \pi_{expensive}(x) = \mu_{expensive}(x)$$

where *expensive* is a linguistic term that represents the fuzzy set of expensive prices.

3. The **veristic constraints**, $r = v$. In such a constraint, X is interpreted as a conjunctive[4] variable —i.e. a variabele which at the same time can contain more than one value of the universe of discourse U—. The semantics of

$$X \ isv \ R$$

with R being a fuzzy set with membership function $\mu_R : U \rightarrow [0,1]$, states that the value of X is the collection of elements of R where the membership grades of R are interpreted as degrees of truth. An example of a veristic constraint is

$$X \ isv \ \{(English, 1.0), (French, 0.8), (German, 0.6)\}$$

which means that X contains the values 'English', 'French' and 'German' with respective truth 1.0, 0.8 and 0.6 (X can for example be a variable that is used to represent the languages spoken by somebody).

4. The **probabilistic constraints**, $r = p$. Hereby, X is seen as a variable whose value is uncertain (in statistical sense). Furthermore, the interpretation of

$$X \ isp \ R$$

is that R is the probability distribution of X. An example of a probabilistic constraint is

$$X \ isp \ N(m, \sigma^2)$$

[3] A disjunctive variable has also been named a possibilistic variable by Zadeh. This is also the origin of the name possibilistic constraint.

[4] A conjunctive variable has also been named a veristic variable by Zadeh. This also clarifies the origin of the name veristic constraint.

which means that the value of X is modelled by a normal probability distribution with mean value m and variance σ^2. Analogously,

$$X\ isp\ \{0.2/a, 0.4/b, 0.4/b\}$$

that the value of X is a, b or c with respective probabilities 0.2, 0.4 and 0.4.

Other, for this book less important, kinds of generalized constraints are the so-called probability value constraints, the arbitrary set constraints and the fuzzy graph constraints [29].

References

1. G. de Cooman, "Towards a possibilistic logic", in: *Fuzzy Set Theory and Advanced Mathematical Applications*, D. Ruan (ed.) (Kluwer Academic Publishers, Boston, USA, 1995) 89–133.
2. G. De Tré, "Extended Possibilistic Truth Values", *International Journal of Intelligent Systems* **17** (2002) 427–446.
3. D. Dubois, and H. Prade, *Fuzzy Sets and Systems: Theory and Applications* (Academic Press, New York, USA, 1980).
4. D. Dubois, and H. Prade, *Possibility Theory* (Plenum Press, New York, USA, 1988).
5. D. Dubois, and H. Prade, "The three semantics of fuzzy sets", *Fuzzy Sets and Systems* **90** 2 (1997) 141–150.
6. D. Dubois, and H. Prade, "Possibility Theory: Qualitative and Quantitative Aspects", in: *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 1* (Kluwer Academic Publishers, The Netherlands, 1998) 169–226.
7. D. Dubois, and H. Prade (eds.), *Fundamentals of fuzzy sets*, The Handbook of Fuzzy Sets Series (Kluwer Academic Publishers, Boston, USA, 2000).
8. B. de Finetti, *Theory of Probability* (Wiley, New York, USA, 1974).
9. G.J. Klir, and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications* (Prentice Hall Inc., New Jersey, USA, 1995).
10. P. Krause, and D. Clark, *Representing Uncertain Knowledge* (Kluwer, Dordrecht, The Netherlands, 1993).
11. E.H. Ruspini, P.P. Bonisonne, and W. Pedrycz (eds.), *Handbook of Fuzzy Computation* (Institute of Physics, 1998).
12. OFTA (ed.), *Logique Floue*, Arago **14** (Masson, Paris, France, 1994).
13. H. Prade, "Possibility sets, fuzzy sets and their relation to Lukasiewicz logic", in: *Proceedings of the 12th International Symposium on Multiple-Valued Logic* (1982) 223–227.
14. N. Rescher, *Many-Valued Logic* (Mc.Graw-Hill, New York, USA, 1969).
15. C.A.B. Smith, "Consistency in Statistical Inference and Decision", *Journal of the Royal Statistical Society* **B23** (1961) 218–258.
16. M. Smithson, *Ignorance and Uncertainty* (Springer, Berlin, Germany, 1989).
17. P. Walley, *Statistical Reasoning with Imprecise Probabilities* (Chapman and Hall, London, England, 1991).
18. P. Walley, "Measures of uncertainty in expert systems", *Artificial Intelligence* **83** (1996) 1–58.
19. S. Weber, "A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorms", *Fuzzy Sets and Systems* **13** 3 (1983) 115–134.
20. R.R. Yager, "On a general class of fuzzy connectives", *Fuzzy Sets and Systems* **4** 3 (1980) 235–242.

21. R.R. Yager, “A characterization of the extension principle”, *Fuzzy Sets and Systems* **18** 3 (1986) 205–217.
22. L.A. Zadeh, “Fuzzy Sets”, *Information and Control* **8** 3 (1965) 338–353.
23. L.A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning I”, *Information Sciences* **8** (1975) 199–251.
24. L.A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning II”, *Information Sciences* **8** (1975) 301–357.
25. L.A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning III”, *Information Sciences* **9** (1975) 43–80.
26. L.A. Zadeh, “Fuzzy sets as a basis for a theory of possibility”, *Fuzzy Sets and Systems* **1** 1 (1978) 3–28.
27. L.A. Zadeh, “A computational approach to fuzzy quantifiers in natural languages”, *Computer Mathematics with Applications* **9** (1983) 149–183.
28. L.A. Zadeh, “Fuzzy Logic = Computing with Words”, *IEEE Transactions on Fuzzy Systems* **4** 2 (1996) 103–111.
29. L.A. Zadeh, “Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic”, *Fuzzy Sets and Systems* **90** 2 (1997) 111–127.
30. L.A. Zadeh, and J. Kacprzyk (eds.), *Computing with Words in Information/Intelligent Systems* (Physica-Verlag, Heidelberg, Germany, 1999)
31. L.A. Zadeh, “A New Direction in AI: Toward a Computational Theory of Perceptions”, *AI Magazine* **22** 1 (2001) 73–84.
32. L.A. Zadeh, “Computing with Words and Perceptions - A Paradigm Shift in Computing and Decision Analysis and Machine Intelligence”, in: *Proceedings of the 2003 International Conference on Machine Learning and Applications - ICMLA 2003* (Los Angeles, USA, 2003).

Chapter 3
Fuzzy querying of regular databases

Databases are a very important component in many information systems. Due to their increasing number and volumes, effective and efficient access to data becomes more and more important. A lot of research has already been done to enhance database accessibility. This research focuses on different aspects like, among others, file organization, indexing, query languages and access techniques.

In this chapter such access techniques, which are based on fuzzy set theory and possibility theory and moreover support flexible database querying, are described in more detail. These techniques will be called **fuzzy database querying techniques**. In this chapter we explicitly assume that the underlying databases are regular databases.

The core idea with fuzzy querying is that flexible preferences are introduced in the query formulations [5]. This can be done at two levels: *inside* elementary query conditions and *between* query conditions. This core idea forms the basis of this chapter. Flexible preferences inside query conditions allow for flexible search criteria which express to what *degree* particular attribute values are adequate for the result. This is further described in section 3.2. Preferences between query conditions are used to associate different grades of importance to different query conditions. This is further handled in section 3.3. To support flexibility of preferences, query languages like SQL and OQL and their underlying algebraic and logical frameworks have been extended. Therefore, among others, more general definitions of the algebraic data manipulation operators have been proposed [3, 28, 4, 29, 21]. Frameworks for fuzzy querying are described in section 3.4.

There exist other flexible querying approaches than the ones described in this chapter. Examples are:

- Querying systems that are able to correct syntactical and semantic errors in the query formulations.
- Querying systems that allow for an intelligent database navigation.
- Querying systems that support ‘indirect’ answers like summaries, conditional answers and background information for (empty) queries [20].

These approaches are not dealt with in this book.

3.1 Example database

Before we start with the description of flexible querying we present an example database that will be used in the remainder of the book to illustrate techniques and concepts. To keep generality, we assume that the database consists of a finite number of **records**. Each record consists of one or more fields that contain data values. Each field is defined by a name and an associated data type[1]; each field value must be an element of the domain[2] of the associated data type. Each record is constructed in accordance with the specifications of its **record type**. A record type is characterized by a name and a set of field definitions. A database is typically stored in one or more record files which are kept in the memory of the database system. Although this is not a requirement, a file can contain records of different record types. In practice the number of records in a database can vary from just some records to millions of records.

Example 3.1
Figure 3.1 contains a representation of the record types and records of the example database for artworks. The database consists of a collection of records of three record types: 'Painting', 'Artist' and 'Owner'. For the sake of readability, the records are ordered by record type and field names are repeated in the heading. ◇

In the example database, the relationship between a painting and a painter is modelled by means of the field 'Artist' of the record type 'Painting' and the field 'Name' of the record type 'Artist'. A 'painting'-record with field value 'Monet' for 'Artist' is hereby associated with the 'artist'-record with the same field value for 'Name'. Whenever this is relevant, the example database will be translated to a relational or object oriented database scheme.

3.2 Fuzzy preferences inside query conditions

In reality it happens that a database user describes only approximately what he or she searches in the database. This could, among others, be due to the fact that the user only has a limited knowledge about what is searched for or be due to the fact that the user explicitly wants to allow some tolerance in the database querying — not only the exact results, but also results that are closely similar are tolerated in the result—.

[1] In a software context, a data type defines the structure and operations of a collection of data with common characteristics.

[2] The domain of a data type is considered to be the set of all the values that are allowed whenever a value of that data type is expected.

RECORDTYPE Painting (ID:**CHAR**(3); Name:**CHAR**(30); Artist:**CHAR**(30); Period:**INTEGER**; Value:**REAL**; Owner:**CHAR**(30))

ID	Name	Artist	Period	Value	Owner
P01	Fishermans house	Monet	1882	16.000.000	Boijmans
P02	The ballet course	Degas	1872	8.500.000	Louvre
P03	Mona Lisa	Da Vinci	1499	75.000.000	Louvre
P04	Afternoon in Ostend	Ensor	1881	200.000	KMSK

RECORDTYPE Artist (Name:**CHAR**(30); First_name:**CHAR**(20); Year_of_birth:**INTEGER**; Year_of_death:**INTEGER**)

Name	First_name	Year_of_birth	Year_of_death
Da Vinci	Leonardo	1452	1519
Degas	Edgar	1834	1917
Ensor	James	1860	1949
Monet	Claude	1840	1926

RECORDTYPE Owner (Name:**CHAR**(30); Place:**CHAR**(20); Country:**CHAR**(20))

Name	Place	Country
Boijmans	Rotterdam	The Netherlands
Louvre	Paris	France
KMSK	Antwerp	Belgium

Fig. 3.1 Records of the example database 'Artworks'.

Example 3.2

- *As an example of the first situation we consider a museum visitor who remembers only that the painting he is looking for dates from 'around 1880'. To find the painting the visitor can query the database for paintings from 'around 1880'.*
- *The second situation occurs for example when a teacher queries his examn database for students with a 'high score'.*

◇

3.2.1 Modelling

3.2.1.1 Linguistic terms

Approximately indicating which values has to be searched for by the querying system, is almost the same as indicating which values are more and which values are less allowable as an answer. This can be modelled by fuzzy sets of which the membership grades are interpreted as *degrees of compatibility*. Such a fuzzy set is thus *conjunctive* by its interpretation. As illustrated in the previous examples, it is more practical to work with **linguistic terms**. In such a case, the linguistic term is used as **label** for the underlying fuzzy set. In accordance with the description of the interpretations of membership grades (cf. previous chapter), the membership grade $\mu_{\tilde{V}}(x)$ of an element x in a fuzzy set $\tilde{V}$ with label L denotes to which extend x is a value that is typically represented by the linguistic term L. If the linguistic term L identifies the fuzzy set $\tilde{V}$, then the membership function $\mu_{\tilde{V}}$ will be denoted by μ_L.

Example 3.3

- *The label 'around 1880' can be modelled by the fuzzy set:*

$$\{(1875,0.2),(1876,0.4),(1877,0.6),(1878,0.8),(1879,1),\\ (1880,1),(1881,1),(1882,0.8),(1883,0.6),(1884,0.4),(1882,0.2)\}$$

- *The label 'high score' can be modelled by the fuzzy set:*

$$\{(16,0.35),(17,0.65),(18,1),(19,1),(20,1)\}$$

$\diamond$

In general, linguistic terms could be associated with fuzzy sets that are defined on several domains $D_1,D_2,\ldots,D_n$ and thus have a membership function of the form $D_1\times D_2\times\cdots\times D_n\rightarrow[0,1]$. Domains could be measurable (numerical) or not.

In practice fuzzy sets that are defined on only one domain are mostly used. If this domain is measurable, then the fuzzy set will mostly be defined by means of a trapezoidal membership function. Discrete membership functions are used in case of unmeasurable domains.

The meaning of a linguistic term and its corresponding fuzzy set could be dependent on the context and on the subjectivity of the user. As such, 'expensive' will for example have a different meaning in the context of 'paintings' than in the context of 'flowers'. Also, the linguistic term 'young' in the context of ages of persons will be completely differently experienced by a child than by an older person.

3.2.1.2 Modification functions

The meaning of a linguistic term can be strengthened or weakened by the use of adverbs. An example of an adverb that strengthens the linguistic term 'expensive' is 'very', whereas 'more or less' is an example of an adverb that weakens the term 'expensive'. The impact of an adverb on a linguistic term can be modelled mathematically by means of so-called **modification functions** which are applied on the corresponding membership function. One can make a distinction between *pre modification* and *post modification*.

In **pre modification** the membership function μ_L is applied onto the modification function m_{pre}. The membership function of the modified linguistic term L^{mod} then becomes

$$\forall\, x\in U:\mu_{L^{mod}}(x)=\mu_L(m_{pre}(x))$$

Pre modification changes the shape and position of the membership function and is therefore most suited in the modelling of vagueness and imprecision.

In **post modification** the modification function m_{post} is applied onto the membership function μ_L. The membership function of the modified linguistic term L^{mod} then becomes

$$\forall\, x \in U : \mu_{L^{mod}}(x) = m_{post}(\mu_L(x))$$

Post modification only changes the shape of the membership function and is therefore most suited in the modelling of uncertainty.

Modification functions that are mostly used to strengthen linguistic terms are the square function ($.^2$) and the fourth power function ($.^4$). The most popular weakening modification functions are the root function ($\sqrt{.}$) and the fourth root function ($.^{1/4}$). Another interesting parametrized modification function, which is used in pre modification and can be used for the modelling of as well strengthening, as weakening adverbs is the so-called twofold, piecewise linear modification function $f_m^{\alpha,\beta}$ which is presented in figure 3.2. The function $f_m^{\alpha,\beta}$ is completely characterized by

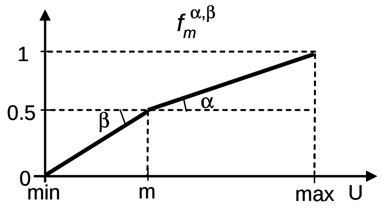

Fig. 3.2 The twofold piecewise linear modification function $f_m^{\alpha,\beta}$.

the parameters m (the domain value that corresponds with membership grade 0.5), β (parameter for the angle of the first linear piece) and α (parameter for the angle of the second linear piece).

The application of post modification and pre modification is illustrated in figure 3.3.

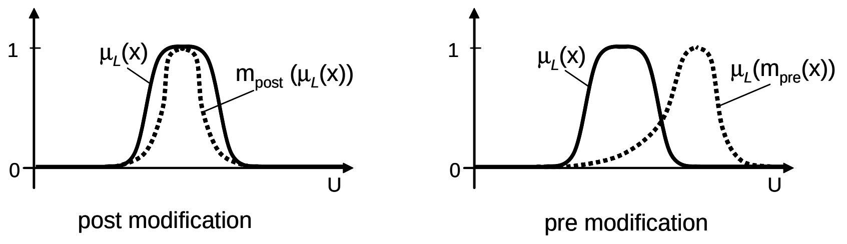

Fig. 3.3 Post and pre modification applied on μ_L.

3.2.1.3 Similarities

An alternative approach, that is especially useful to deal with unmeasurable domains, is to work with similarities. To model the compatibility between two domain values a grade of similarity is associated with any pair of domain values. Again, one can work with grades that are elements of the unit interval $[0,1]$. Formally, similarity is then characterized by a so-called **similarity relation** which is defined as a binary fuzzy relation

$$S : U \times U \rightarrow [0,1]$$

that satisfies the following axioms:

- Axiom s1. $\forall\, x \in U : S(x,x) = 1$ (reflexivity).
- Axiom s2. $\forall\, (x,y) \in U^2 : S(x,y) = S(y,x)$ (symmetry).

Optionally one can also consider a transitivity condition:

- Axiom s3. $\forall\, (x,z) \in U^2 : S(x,z) \geq \sup_{y \in U} \min(S(x,y), S(y,z))$ (sup-min transitivity).

Given a domain value $x \in U$ and a similarity relation S, it is always possible to construct a fuzzy set of compatible values of x. The membership grades of this fuzzy set are interpreted as degrees of compatibility. The domain value could be used to label the fuzzy set. The fuzzy set $\tilde{V}_x$, representing (the similar values of) the domain value $x \in U$ could be obtained as follows:

$$\tilde{V}_x = \{(y, S(x,y)) | y \in U\}$$

Departing from such a fuzzy set, we can search for appropriate modification functions to model the impact of strengthening and weakening adverbs. In this way strengthening and weakening adverbs could be handled in the case of unmeasurable domains.

Example 3.4
The following similarity relation S could be defined for the linguistic terms 'green', 'light green', 'kaki green' and 'light brown':

S	*green*	*light green*	*kaki green*	*light brown*
green	*1*	*0.8*	*0.6*	*0.2*
light green	*0.8*	*1*	*0.8*	*0.4*
kaki green	*0.6*	*0.8*	*1*	*0.6*
light brown	*0.2*	*0.4*	*0.6*	*1*

On the basis of this similarity relation we can build the following fuzzy sets:

$$\begin{aligned}
\mathit{green} &= \{(\mathit{green}, 1), (\mathit{light\ green}, 0.8), (\mathit{kaki\ green}, 0.6), (\mathit{light\ brown}, 0.2)\}\\
\mathit{light\ green} &= \{(\mathit{green}, 0.8), (\mathit{light\ green}, 1), (\mathit{kaki\ green}, 0.8), (\mathit{light\ brown}, 0.4)\}\\
\mathit{kaki\ green} &= \{(\mathit{green}, 0.6), (\mathit{light\ green}, 0.8), (\mathit{kaki\ green}, 1), (\mathit{light\ brown}, 0.6)\}\\
\mathit{light\ brown} &= \{(\mathit{green}, 0.2), (\mathit{light\ green}, 0.4), (\mathit{kaki\ green}, 0.6), (\mathit{light\ brown}, 1)\}
\end{aligned}$$

◇

3.2.2 *Evaluation*

Considering flexible search criteria in query formulation brings along with it that a database object could only partially satisfy the criteria. This cannot be adequately handled by Boolean logic. Therefore there is a need for a framework that is based on a many-valued logic. In this book we describe two such frameworks: a simple approach that is based on fuzzy logic and fuzzy sets where membership grades are interpreted as degrees of truth and a more advanced approach that is based on (extended) possibilistic truth values. In both frameworks, the evaluation of a search condition can be seen as the comparison of a value that is retrieved from the database and the collection of allowed values that are specified in the query formulation.

3.2.2.1 Approaches with satisfaction degrees

In these approaches, the evaluation of a query condition results for each database object in an associated degree of truth. These degrees of truth are interpreted as **satisfaction degrees** and are calculated by means of evaluation and aggregation functions. Usually only objects with an associated degree that differs from zero are considered in the result set of the query.

Evaluation of simple conditions.

In fuzzy querying, a query can contain both regular and flexible query conditions. For the evaluation of a simple (or atomic) regular condition, classical Boolean logic can be used. If the condition evaluates to true (T), then the corresponding satisfaction degree should equal to 1, denoting that the database entry under consideration completely satisfies the condition. Otherwise, if the condition evaluates to false (F), then the corresponding satisfaction degree should be equal to 0, denoting that database entry under consideration does not satisfy the condition at all.

The best known evaluation functions for simple (or atomic) fuzzy query conditions are the comparison operators and the compatibility operator.

- **Comparison operators ($=$, $\neq$, $<$, $\leq$, $>$, $\geq$).**
  - To model the equality operator $=$, Zadeh's standard equality operator (cf. definition 2.11) or the gradual equality operator (cf. definition 2.12) can be used. Hereby, the regular attribute values a stored in the database need to be converted to their corresponding fuzzy sets $\{(a,1)\}$.
    - · When using Zadeh's standard equality operator:
      - · Equality corresponds with a satisfaction degree 1;

    · Inequality corresponds with a satisfaction degree 0.
  · When using the gradual equality operator, the satisfaction degree corresponds to the resulting degree of equality.
- The comparison operators $<$, $>$, $\leq$ and $\geq$ can for example be obtained by applying Zadeh's extension principle (cf. definition 2.15). Again, the regular attribute values a stored in the database must be converted to their corresponding fuzzy sets $\{(a,1)\}$. Given that $op \in \{<,>,\leq,\geq\}$ and L being the fuzzy set of acceptable values, the satisfaction degree is in case of the operator op obtained by
$$\underset{\{x|x\in U \wedge a\ op\ x\}}{sup} \quad \mu_L(x)$$

- **Compatibility operator (IS).** In most cases a user will be rather interested in the extent to which an attribute value a that is retrieved from a database *is compatible with* the allowed values that are specified in the query condition. In the simple approach under consideration this degree of compatibility is considered to be the membership grade $\mu_L(a)$ of a in the fuzzy set L of acceptable values that is specified in the query condition. Thus, if A is a database attribute and a is the value of A in the record r, then the evaluation $e(A\ IS\ L)(r)$ of $A\ IS\ L$ for r is defined by
$$e(A\ IS\ L)(r) = \mu_L(a).$$

An even more flexible way to deal with comparison operators is to use 'fuzzy' comparison operators. 'Fuzzy' comparison operators can be modelled by means of a membership function μ_{op} which is defined over the Cartesian product of two domains and which denotes for each couple of domain values to which extent the operator is satisfied. In this way, we can for example define operators like 'approximately equal to' and 'much larger than'. For the evaluation of a 'fuzzy' comparison operator the same method as with the evaluation of the compatibility operator can be used. Assume that op represents the 'fuzzy' comparison operator, that a is the value of attribute A that is stored in the database record r and that L is the fuzzy set of allowed allowed values for A with membership function μ_L that is specified in the query condition. With these assumptions, the satisfaction degree $e(A\ op\ L)(r)$ of $A\ op\ L$ for r is obtained as

$$e(A\ op\ L)(r) = \mu_{L\,\circ\,op}(a)$$

where
$$\mu_{L\,\circ\,op}(a) = \sup_{x\in dom_A} \min(\mu_{op}(a,x), \mu_L(x)).$$

Evaluation of composite conditions.

Generally, a query condition consists of a logical expression that is composed of simple conditions, logical operators (conjunction, disjunction and negation) and brackets that are used to enforce precedencies on logical operators. The evaluation of a composite condition is completely analogously as in Boolean logic. Firstly, the simple conditions are evaluated (each evaluation results in a satisfaction degree). After that, the expression is evaluated. Brackets, if existent, are appropriately dealt with. Negation ($\neg$) has precedency over conjunction ($\wedge$) and disjunction ($\vee$), whereas conjunction ($\wedge$) has precedency over disjunction ($\vee$). The evaluation of an expression is from left to right, unless brackets are used to enforce precedency.

The conjunction, disjunction and negation operators for satisfaction degrees are defined as follows:

- **Conjunction.** $\mu_{p \wedge q} = \min(\mu_p, \mu_q)$.
- **Disjunction.** $\mu_{p \vee q} = \max(\mu_p, \mu_q)$.
- **Negation.** $\mu_{\neg(p)} = 1 - \mu_p$.

Remark that other definitions for conjunction and disjunction can be used. In fact, it is sufficient to replace the couple $(\min, \max)$ by another (t-norm, t-conorm)-couple (i, u).

3.2.2.2 Approaches with (extended) possibilistic truth values

Instead of using satisfaction degrees, one can also work with (extended) possibilistic truth values ((E)PTV's) to model query satisfaction. In such a case the definition of a record type is considered to be a predicate. The records of the record type are then propositions which for regular database querying on regular databases all have to evaluate to true (T). When considering fuzzy querying the truth values of these propositions could be gradual. To belong to the record set, the truth value of the proposition of a record must differ from false (F).

(Extended) possibilistic truth values can be used to model a gradation of truth. An (E)PTV then expresses to which extent it is *possible* (or *impossible*) that the proposition under consideration is satisfied. In an approach with (E)PTV's the evaluation of a query condition always results in an **(extended) possibilistic truth value**. Here also, query conditions are evaluated using appropriate evaluation functions.

Evaluation of simple conditions.

For the evaluation of simple (or atomic) regular conditions that are part of a fuzzy query formulation classical Boolean logic can again be used. If a condition evaluates to true (T), then the corresponding (E)PTV is $\{(T, 1)\}$, which denotes that it is certain that the database entry under consideration completely satisfies the condition. On the other hand, if the condition evaluates to false (F), then the corresponding (E)PTV is $\{(F, 1)\}$, denoting that for the considered database entry, the condition is not satisfied at all.

For simple (or atomic) fuzzy query conditions the comparison operators and compatibility operator are modelled as follows.

- **Comparison operators ($=$, $\neq$, $<$, $\leq$, $>$, $\geq$).**

– The modelling of the equality operator $=$ can again be based on Zadeh's standard equality operator (cf. definition 2.11) or be based on the gradual equality operator (cf. definition 2.12). Again, the regular attribute values a stored in the database must be converted to their fuzzy counterpart $\{(a,1)\}$.
  · When using Zadeh's standard equality operator:
    · Equality corresponds with an (E)PTV $\{(T,1)\}$;
    · Inequality corresponds with an (E)PTV $\{(F,1)\}$.
  · When using the gradual equality operator, the resulting (E)PTV is computed using the obtained degree of equality deg. This is done as follows:

$$\{(T, \frac{\text{deg}}{\max(\text{deg}, 1-\text{deg})}), (F, \frac{1-\text{deg}}{\max(\text{deg}, 1-\text{deg})})\}$$

.

– The comparison operators $<$, $>$, $\leq$ and $\geq$ can in a framework with EPTV's also be obtained by applying Zadeh's extension principle (cf. definition 2.15). The regular attribute values a stored in the database are first converted to their corresponding fuzzy set $\{(a,1)\}$. With $op \in \{<,>,\leq,\geq\}$ and L being the fuzzy set of acceptable values, the (E)PTV is in case of the operator op then subsequently computed by

$$\{(T, \frac{\mu_T}{\max(\mu_T, \mu_F)}), (F, \frac{\mu_F}{\max(\mu_T, \mu_F)})\}$$

where

$$\mu_T = \sup_{\{x|x\in U \wedge a\ op\ x\}} \mu_L(x)$$

and

$$\mu_F = \sup_{\{x|x\in U \wedge NOT\,(a\ op\ x)\}} \mu_L(x)$$

- **Compatibility operator (*IS*).** The degree of compatibility between a stored value a of attribute A in the record r and a given fuzzy set L of acceptable values, specified in the query condition, thus the result of the evaluation $e(A\ IS\ L)(r)$, is obtained by

$$e(A\ IS\ L)(r) = \{(T, \frac{\mu_L(a)}{\max(\mu_L(a), 1-\mu_L(a))}), (F, \frac{1-\mu_L(a)}{\max(\mu_L(a), 1-\mu_L(a))})\}.$$

'Fuzzy' comparison operators can also in this framework be used as a more flexible alternative for the modelling of comparison operators. Assume that op represents the 'fuzzy' comparison operator that is modelled by the membership function μ_{op} which is defined over the Cartesian product of two domains and which denotes for each couple of domain values the extent to which the operator is satisfied. Furthermore, assume that a is the value that is stored for attribute A in the database record r and that L is the fuzzy set of allowed values for A with membership function μ_L which is specified in the query condition. With these assumptions, the (E)PTV $e(A\ op\ L)(r)$ of $A\ op\ L$ for r is obtained —in a similar way as in the approaches with satisfaction degrees— as follows:

$$e(A\,op\,L)(r) = \{(T, \frac{\mu_{L\,\circ\,op}(a)}{\max(\mu_{L\,\circ\,op}(a), 1-\mu_{L\,\circ\,op}(a))}), \\ (F, \frac{1-\mu_{L\,\circ\,op}(a)}{\max(\mu_{L\,\circ\,op}(a), 1-\mu_{L\,\circ\,op}(a))})\}$$

where

$$\mu_{L\circ op}(a) = \sup_{x\in dom_A} \min(\mu_{op}(a,x), \mu_L(x)).$$

Evaluation of composite conditions.

The evaluation of composite conditions is analogous as in Boolean logic. The evaluation starts with the evaluation of the simple conditions (each evaluation results in an (E)PTV's). Subsequently, the composite condition is evaluated. Brackets, if existent, are appropriately dealt with.

For the conjunction, disjunction and negation of (E)PTV's we can use the computation rules that are given in section 2.4.2.2. Alternative computation rules can be used [13]. Below, two alternatives for the conjunction operator for possibilistic truth values are described. Both alternatives are based on a (t-norm,t-conorm)-couple (i,u) and can be generally defined as follows (with $I = \{T,F\}$):

$$\tilde{\wedge}_i : \tilde{\wp}(I) \times \tilde{\wp}(I) \to \tilde{\wp}(I) : (\tilde{U},\tilde{V}) \mapsto \tilde{U}\ \tilde{\wedge}_i\ \tilde{V}$$

where:

- $\mu_{\tilde{U}\tilde{\wedge}_i\tilde{V}}(T) = i(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(T))$
- $\mu_{\tilde{U}\tilde{\wedge}_i\tilde{V}}(F) = u(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(F))$

This definition reflects that the membership grade $\mu_{\tilde{U}\tilde{\wedge}_i\tilde{V}}(T)$ is mostly 'influenced' by the worst (the smallest) of the membership grades $\mu_{\tilde{U}}(T)$ and $\mu_{\tilde{V}}(T)$; furthermore it is also reflected that the membership grade $\mu_{\tilde{U}\tilde{\wedge}_i\tilde{V}}(F)$ is mostly 'influenced' by the worst (the largest) of the membership grades $\mu_{\tilde{U}}(F)$ and $\mu_{\tilde{V}}(F)$.

Proposition 2 *All operators $\tilde{\wedge}_i$ are truth functional, i.e. $\forall\ p,q \in P : \tilde{t}(p\ \wedge\ q) = \tilde{t}(p)\ \tilde{\wedge}_i\ \tilde{t}(q)$.*

Proof:
This proposition is proved by a full case study of the modelling of the classical logical conjunction:

i. If $\tilde{t}(p) = \{(T,1)\}$ and $\tilde{t}(q) = \{(T,1)\}$, then it must hold that

$$\tilde{t}(p\ \wedge\ q) = \{(T,1)\}$$

$$\begin{aligned}\tilde{t}(p)\ \tilde{\wedge}_i\ \tilde{t}(q) &= \{(T,i(1,1)),(F,u(0,0))\}\\ &= \{(T,1)\}\ \textit{(Axioms i1 and u1.)}\\ &= \tilde{t}(p\wedge q)\end{aligned}$$

ii. If $\tilde{t}(p)=\{(T,1)\}$ and $\tilde{t}(q)=\{(F,1)\}$, then it must hold that

$$\tilde{t}(p\wedge q)=\{(F,1)\}$$

$$\begin{aligned}\tilde{t}(p)\ \tilde{\wedge}_i\ \tilde{t}(q) &= \{(T,i(1,0)),(F,u(0,1))\}\\ &= \{(F,1)\}\ \textit{(Axioms i1, i3, u1 and u3.)}\\ &= \tilde{t}(p\wedge q)\end{aligned}$$

iii. If $\tilde{t}(p)=\{(F,1)\}$ en $\tilde{t}(q)=\{(T,1)\}$, then it must hold that

$$\tilde{t}(p\wedge q)=\{(F,1)\}$$

$$\begin{aligned}\tilde{t}(p)\ \tilde{\wedge}_i\ \tilde{t}(q) &= \{(T,i(0,1)),(F,u(1,0))\}\\ &= \{(T,1)\}\ \textit{(Axioms i1, i3, u1 and u3.)}\\ &= \tilde{t}(p\wedge q)\end{aligned}$$

iv. If $\tilde{t}(p)=\{(F,1)\}$ and $\tilde{t}(q)=\{(F,1)\}$, then it must hold that

$$\tilde{t}(p\wedge q)=\{(F,1)\}$$

$$\begin{aligned}\tilde{t}(p)\ \tilde{\wedge}_i\ \tilde{t}(q) &= \{(T,i(0,0)),(F,u(1,1))\}\\ &= \{(F,1)\}\ \textit{(Axioms i2, i1, u2 and u1.)}\\ &= \tilde{t}(p\wedge q)\end{aligned}$$

$\circ$

To obtain concrete definitions of conjunctive aggregation operators, we must choose a (t-norm,t-conorm)-couple. Two examples are:

- Zadeh t-norm and t-conorm $(\min,\max)$:

$$\tilde{\wedge}_{Zadeh} : \tilde{\wp}(I)\times\tilde{\wp}(I)\to\tilde{\wp}(I) : (\tilde{U},\tilde{V})\mapsto\tilde{U}\ \tilde{\wedge}_{Zadeh}\ \tilde{V}$$

  where:
$$\mu_{\tilde{U}\tilde{\wedge}_{Zadeh}\tilde{V}}(T)=\min(\mu_{\tilde{U}}(T),\mu_{\tilde{V}}(T))$$
  and
$$\mu_{\tilde{U}\tilde{\wedge}_{Zadeh}\tilde{V}}(F)=\max(\mu_{\tilde{U}}(F),\mu_{\tilde{V}}(F))$$

- Probabilistisc t-norm and t-conorm $(ab, a+b-ab)$:

$$\tilde{\wedge}_{prob} : \tilde{\wp}(I) \times \tilde{\wp}(I) \to \tilde{\wp}(I) : (\tilde{U}, \tilde{V}) \mapsto \tilde{U} \; \tilde{\wedge}_{prob} \; \tilde{V}$$

where:

$$\mu_{\tilde{U}\tilde{\wedge}_{prob}\tilde{V}}(T) = \mu_{\tilde{U}}(T).\mu_{\tilde{V}}(T)$$

and

$$\mu_{\tilde{U}\tilde{\wedge}_{prob}\tilde{V}}(F) = \mu_{\tilde{U}}(F) + \mu_{\tilde{V}}(F) - \mu_{\tilde{U}}(F).\mu_{\tilde{V}}(F)$$

Because of proposition 2, both alternatives are truth functional.

Analogously, we can define alternative computation rules for the disjunction. For a (t-norm,t-conorm)-couple (i,u) and $I = \{T,F\}$ one can find that:

$$\tilde{\vee}_{u} : \tilde{\wp}(I) \times \tilde{\wp}(I) \to \tilde{\wp}(I) : (\tilde{U}, \tilde{V}) \mapsto \tilde{U} \; \tilde{\vee}_{u} \; \tilde{V}$$

where:

- $\mu_{\tilde{U}\tilde{\vee}_{u}\tilde{V}}(T) = u(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(T))$
- $\mu_{\tilde{U}\tilde{\vee}_{u}\tilde{V}}(F) = i(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(F))$

Concrete definitions of disjunctive aggregation operators are then obtained by choosing a (t-norm,t-conorm)-couple. Two examples:

- Zadeh t-norm and t-conorm $(\min, \max)$:

$$\tilde{\vee}_{Zadeh} : \tilde{\wp}(I) \times \tilde{\wp}(I) \to \tilde{\wp}(I) : (\tilde{U}, \tilde{V}) \mapsto \tilde{U} \; \tilde{\vee}_{Zadeh} \; \tilde{V}$$

  where:

$$\mu_{\tilde{U}\tilde{\vee}_{Zadeh}\tilde{V}}(T) = \max(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(T))$$

  and

$$\mu_{\tilde{U}\tilde{\vee}_{Zadeh}\tilde{V}}(F) = \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(F))$$

- Probabilistic t-norm and t-conorm $(ab, a+b-ab)$:

$$\tilde{\vee}_{prob} : \tilde{\wp}(I) \times \tilde{\wp}(I) \to \tilde{\wp}(I) : (\tilde{U}, \tilde{V}) \mapsto \tilde{U} \; \tilde{\vee}_{prob} \; \tilde{V}$$

  where:

$$\mu_{\tilde{U}\tilde{\vee}_{prob}\tilde{V}}(T) = \mu_{\tilde{U}}(T) + \mu_{\tilde{V}}(T) - \mu_{\tilde{U}}(T).\mu_{\tilde{V}}(T)$$

  and

$$\mu_{\tilde{U}\tilde{\vee}_{prob}\tilde{V}}(F) = \mu_{\tilde{U}}(F).\mu_{\tilde{V}}(F)$$

These alternative definitions for conjunction and disjunction have, among others, as advantage that they are easier to implement and thus better suited for fuzzy querying applications. A more elaborated discussion can be found in [14].

3.3 Preferences between query conditions

Traditionally all simple conditions in a composite condition are considered to be of equal importance. Nevertheless, it can occur that some conditions are more important (or more relevant) for the user than others so that the impact of their evaluation on the query result must be larger than others. This means in fact that the user puts preferences between query conditions. Putting preferences between query conditions is in most approaches realized by associating **weights** with the conditions.

Example 3.5
A visitor of the museum wants an overview of all paintings of Degas that are painted 'around 1870'. Hereby, it is for the user more important that the painting is a Degas than that the period matches. ◇

3.3.1 Modelling

Associating weights with query conditions can be modelled by coupling a real number, taken from the unit interval $[0,1]$, to each condition. To obtain a semantic meaningful interpretation of weights, these numbers must satisfy the following conditions [17]:

Assume that $w_i \in [0,1]$ is the weight that is associated with condition c_i, where $w_i = 0$ denotes 'totally not important' and $w_i = 1$ denotes 'totally important'.

- In order to have a suited reference and scaling, it must hold that

$$\max_i w_i = 1.$$

- If $w_i = 1$ and c_i evaluates to 'false' (F), then the impact of the weight must be equal to 'false' (F). With other words, if c_i is fully satisfied and c_i is totally important, then the weight may not change anything to the satisfaction of c_i.
- If $w_i = 1$ and c_i evaluates to 'true' (T), then the impact of the weight must be equal to 'true' (T). With other words, if c_i is fully satisfied and c_i is totally important, then the weight may not change anything to the satisfaction of c_i.
- Finally, if $w_i = 0$ then the impact of the weight must result in the neutral element. The neutral element of the conjunction ($\wedge$) is 'true' (T), the neutral element of the disjunction ($\vee$) is 'false' (F). With other words, if c_i is totally not important, then the impact of the weight must be such that it seems like c_i does not exist.

3.3.2 Evaluation

To adequately model the impact of weights within the framework that is used to support fuzzy querying, implication operators can be used. The two kinds of approaches described in section 3.2 are further developed in the remainder of this subsection.

3.3.2.1 Approaches with satisfaction degrees

As described in section 3.2 in an approach with satisfaction degrees, the evaluation of a query condition results in a satisfaction degree ($\in [0,1]$). A possible way to deal with weights is as follows:

1. Evaluate the simple conditions as if there are no weights. This results for each simple condition in a satisfaction degree. The weight of the simple condition is then associated with this satisfaction degree.
2. Consider an implication operator $\overset{\wedge}{\Rightarrow}_f$ (resp. $\overset{\vee}{\Rightarrow}_f$) for conjunction (resp. disjunction).
3. Define an extended version $\wedge^w{}_i$ (resp. $\vee^w{}_u$) of the conjunction operator $\wedge$ (resp. disjunction operator $\vee$). Hereby, the implication operator $\overset{\wedge}{\Rightarrow}_f$ (resp. $\overset{\vee}{\Rightarrow}_f$) is used. The extended operators take as arguments the satisfaction degrees that must be aggregated and their associated weights and result in an aggregated satisfaction degree with an associated aggregated weight.
4. Define an extended version $\neg^w$ of the negation operator $\neg$. This extended operator takes as arguments a satisfaction degree and its associated weight and results in the negation of the satisfaction degree and the same (unmodified) associated weight.
5. Evaluate the composite query condition, hereby using the extended operators for conjunction, disjunction and negation ($\wedge^w{}_i$, $\vee^w{}_u$ and $\neg^w$).

For the definition of the implication operators for conjunction $\overset{\wedge}{\Rightarrow}_f$ and disjunction $\overset{\vee}{\Rightarrow}_f$ a fuzzy implication $\Rightarrow_f$, as introduced in subsection 2.1.4.3, can be used. In general, we then obtain as:

- Implicator operator for conjunction

$$\begin{aligned}\overset{\wedge}{\Rightarrow}_f\colon [0,1]\times[0,1] &\to [0,1]\\ (w,v) &\mapsto (w \Rightarrow_f v)\end{aligned}$$

- Implicator operator for disjunction

$$\begin{aligned}\overset{\vee}{\Rightarrow}_f\colon [0,1]\times[0,1] &\to [0,1]\\ (w,v) &\mapsto \neg(w \Rightarrow_f \neg(v))\end{aligned}$$

Implicator operators that are useful in practice are:

- *Kleene-Dienes*: $(P \Rightarrow_{K-D} Q) = (\max(1-p,q))$
- *Reichenbach*: $(P \Rightarrow_{Rb} Q) = (1-p+p.q)$
- *Gödel*: $(P \Rightarrow_{Go} Q) = \begin{cases} 1 & \text{als } p \leq q \\ q & \text{anders} \end{cases}$

The extended versions of the operators for conjunction ($\wedge^{w}{}_{i}$), disjunction ($\vee^{w}{}_{u}$) and negation ($\neg^{w}$) can be defined as follows:

- Extended operator for conjunction

$$\wedge^{w}{}_{i} : ([0,1]\times[0,1])^2 \to [0,1]\times[0,1]$$
$$((w_1,v_1),(w_2,v_2)) \mapsto (u(w_1,w_2), i(\overset{\wedge}{\Rightarrow}_f (w_1,v_1), \overset{\wedge}{\Rightarrow}_f (w_2,v_2)))$$

- Extended operator for disjunction

$$\vee^{w}{}_{u} : ([0,1]\times[0,1])^2 \to [0,1]\times[0,1]$$
$$((w_1,v_1),(w_2,v_2)) \mapsto (u(w_1,w_2), u(\overset{\vee}{\Rightarrow}_f (w_1,v_1), \overset{\vee}{\Rightarrow}_f (w_2,v_2)))$$

- Extended operator for negation

$$\neg^{w} : [0,1]\times[0,1] \to [0,1]\times[0,1]$$
$$(w,v) \mapsto (w, 1-v)$$

With the choice for using a t-conorm in the computation of the weights in the definitions of the extended operators for conjunction and disjunction it is modelled that the importance of a conjunction (resp. disjunction) can not be less than the importance than its arguments.

3.3.2.2 Approaches with (extended) possibilistic truth values

In these approaches, the evaluation of a query condition results in an (E)PTV (cf. 3.2). Weights can be modelled, completely analogously as in the approaches with satisfaction degrees, by:

1. Evaluate the simple conditions as if there are no weights. This results for each simple condition in an (E)PTV. The weight of the simple condition is then associated with this (E)PTV.
2. Consider an implication operator $\overset{\wedge}{\Rightarrow}_f$ (resp. $\overset{\vee}{\Rightarrow}_f$) for conjunction (resp. disjunction).
3. Define an extended version $\wedge^{w}{}_{i}$ (resp. $\vee^{w}{}_{u}$) of the conjunction operator $\wedge$ (resp. disjunction operator $\vee$). Hereby, the implication operator $\overset{\wedge}{\Rightarrow}_f$ (resp. $\overset{\vee}{\Rightarrow}_f$) is used. The extended operators take as arguments the (E)PTVs that must be aggregated and their associated weights and result in an aggregated (E)PTV with an associated aggregated weight.

4. Define an extended version $\neg^w$ of the negation operator $\neg$. This extended operator takes as arguments an (E)PTV and its associated weight and results in the negation of the (E)PTV and the same (unmodified) associated weight.
5. Evaluate the composite query condition, hereby using the extended operators for conjunction, disjunction and negation ($\wedge^w{}_i$, $\vee^w{}_u$ and $\neg^w$).

For the definition of the implication operators for conjunction $\overset{\wedge}{\Rightarrow}_f$ and disjunction $\overset{\vee}{\Rightarrow}_f$ the fuzzy implication operator $\Rightarrow_f$, as introduced in subsection 2.1.4.3, can be used. In general, with $I = \{T, F\}$ we then obtain:

- Implicator operator for conjunction

$$\begin{aligned}\overset{\wedge}{\Rightarrow}_f \colon [0,1] \times \tilde{\wp}(I) &\to \tilde{\wp}(I)\\ (w, \tilde{V}) &\mapsto \overset{\wedge}{\Rightarrow}_f (w, \tilde{V})\end{aligned}$$

where

$$\mu_{\overset{\wedge}{\Rightarrow}_f (w,\tilde{V})}(T) = (w \Rightarrow_f \mu_{\tilde{V}}(T))$$

and

$$\mu_{\overset{\wedge}{\Rightarrow}_f (w,\tilde{V})}(F) = \neg(w \Rightarrow_f \neg(\mu_{\tilde{V}}(F)))$$

- Implicator operator for disjunction

$$\begin{aligned}\overset{\vee}{\Rightarrow}_f \colon [0,1] \times \tilde{\wp}(I) &\to \tilde{\wp}(I)\\ (w, \tilde{V}) &\mapsto \overset{\vee}{\Rightarrow}_f (w, \tilde{V})\end{aligned}$$

where

$$\mu_{\overset{\vee}{\Rightarrow}_f (w,\tilde{V})}(T) = \neg(w \Rightarrow_f \neg(\mu_{\tilde{V}}(T)))$$

and

$$\mu_{\overset{\vee}{\Rightarrow}_f (w,\tilde{V})}(F) = (w \Rightarrow_f \mu_{\tilde{V}}(F))$$

Implication operators that are interesting in practice are also here:

- *Kleene-Dienes*: $(P \Rightarrow_{K-D} Q) = (\max(1-p, q))$
- *Reichenbach*: $(P \Rightarrow_{Rb} Q) = (1 - p + p.q)$
- *Gödel*: $(P \Rightarrow_{Go} Q) = \begin{cases} 1 & \text{als } p \leq q \\ q & \text{anders} \end{cases}$

The extended versions of the conjunction ($\wedge^w{}_i$), disjunction ($\vee^w{}_u$) and negation ($\neg^w$) operators can be defined as follows:

- Extended operator for conjunction

$$\begin{aligned}\wedge^w{}_i : ([0,1] \times \tilde{\wp}(I))^2 &\to [0,1] \times \tilde{\wp}(I)\\ ((w_1, \tilde{V}_1), (w_2, \tilde{V}_2)) &\mapsto (u(w_1, w_2), \overset{\wedge}{\Rightarrow}_f (w_1, \tilde{V}_1) \tilde{\wedge}_i \overset{\wedge}{\Rightarrow}_f (w_2, \tilde{V}_2))\end{aligned}$$

where $\tilde{\wedge}_i$ is a conjunction operator for possibilistic truth values —based on a (t-norm,t-conorm)-couple (i,u)—, as described in subsection 3.2.2.2.

- Extended operator for disjunction

$$\begin{aligned}\vee^w{}_u : ([0,1] \times \tilde{\wp}(I))^2 &\to [0,1] \times \tilde{\wp}(I)\\ ((w_1,\tilde{V}_1),(w_2,\tilde{V}_2)) &\mapsto (u(w_1,w_2), \overset{\vee}{\Rightarrow}_f (w_1,\tilde{V}_1) \tilde{\vee}_u \overset{\vee}{\Rightarrow}_f (w_2,\tilde{V}_2)))\end{aligned}$$

where $\tilde{\vee}_u$ is a disjunction operator for possibilistic truth values —based on a (t-norm,t-conorm)-couple (i,u)—, as described in subsection 3.2.2.2.

- Extended operator for negation

$$\begin{aligned}\neg^w : [0,1] \times \tilde{\wp}(I) &\to [0,1] \times \tilde{\wp}(I)\\ (w,\tilde{V}) &\mapsto (w, \tilde{\neg}(\tilde{V}))\end{aligned}$$

where $\tilde{\neg}$ is the negation operator for possibilistic truth values, as described in subsection 2.4.2.1.

With the choice for using a t-conorm in the computation of the weights in the definitions of the extended operators for conjunction and disjunction it is also in these frameworks modelled that the importance of a conjunction (resp. disjunction) can not be less than the importance than its arguments.

Example 3.6
As an example, the definitions introduced above are worked out for the case where Gödel's implicator operator is used, i.e.

$$(P \Rightarrow_{Go} Q) = \begin{cases} 1 & \text{iff } p \leq q \\ q & \text{elsewhere} \end{cases}$$

With this operator, the implication operator for conjunction becomes:

$$\begin{aligned}\overset{\wedge}{\Rightarrow}_{Go} : [0,1] \times \tilde{\wp}(I) &\to \tilde{\wp}(I)\\ (w,\tilde{V}) &\mapsto \overset{\wedge}{\Rightarrow}_{Go} (w,\tilde{V})\end{aligned}$$

where

- $\mu_{\overset{\wedge}{\Rightarrow}_{Go}(w,\tilde{V})}(T) = \begin{cases} 1 & \text{iff } w \leq \mu_{\tilde{V}}(T) \\ \mu_{\tilde{V}}(T) & \text{elsewhere} \end{cases}$
- $\mu_{\overset{\wedge}{\Rightarrow}_{Go}(w,\tilde{V})}(F) = \begin{cases} 0 & \text{iff } 1-w \geq \mu_{\tilde{V}}(F) \\ \mu_{\tilde{V}}(F) & \text{elsewhere} \end{cases}$

The implication operator for disjunction becomes:

$$\begin{aligned}\overset{\vee}{\Rightarrow}_{Go} : [0,1] \times \tilde{\wp}(I) &\to \tilde{\wp}(I)\\ (w,\tilde{V}) &\mapsto \overset{\vee}{\Rightarrow}_{Go} (w,\tilde{V})\end{aligned}$$

where

- $\mu_{\overset{\vee}{\Rightarrow}_{Go}(w,\tilde{V})}(T) = \begin{cases} 0 & \textit{iff } 1-w \geq \mu_{\tilde{V}}(T) \\ \mu_{\tilde{V}}(T) & \textit{elsewhere} \end{cases}$
- $\mu_{\overset{\vee}{\Rightarrow}_{Go}(w,\tilde{V})}(F) = \begin{cases} 1 & \textit{iff } w \leq \mu_{\tilde{V}}(F) \\ \mu_{\tilde{V}}(F) & \textit{elsewhere} \end{cases}$

Using the probabilistic t-norm and t-conorm

$$(ab, a+b-ab)$$

the extended versions of the conjunction ($\wedge^{w}{}_{prob}$), disjunction ($\vee^{w}{}_{prob}$) and negation ($\neg^{w}$) operators are defined as follows:

- *Extended operator for conjunction*

$$\begin{aligned} \wedge^{w}{}_{prob} : ([0,1] \times \tilde{\wp}(I))^2 &\to [0,1] \times \tilde{\wp}(I) \\ ((w_1,\tilde{V}_1),(w_2,\tilde{V}_2)) &\mapsto (w_1 + w_2 - w_1.w_2, \\ &\qquad (\overset{\triangle}{\Rightarrow}_{Go}\ (w_1,\tilde{V}_1))\ \tilde{\wedge}_{prob}\ (\overset{\triangle}{\Rightarrow}_{Go}\ (w_2,\tilde{V}_2))) \end{aligned}$$

- *Extended operator for disjunction*

$$\begin{aligned} \vee^{w}{}_{prob} : ([0,1] \times \tilde{\wp}(I))^2 &\to [0,1] \times \tilde{\wp}(I) \\ ((w_1,\tilde{V}_1),(w_2,\tilde{V}_2)) &\mapsto (w_1 + w_2 - w_1.w_2, \\ &\qquad (\overset{\vee}{\Rightarrow}_{Go}\ (w_1,\tilde{V}_1))\ \tilde{\vee}_{prob}\ (\overset{\vee}{\Rightarrow}_{Go}\ (w_2,\tilde{V}_2))) \end{aligned}$$

- *Extended operator for negation*

$$\begin{aligned} \neg^{w} : [0,1] \times \tilde{\wp}(I) &\to [0,1] \times \tilde{\wp}(I) \\ (w,\tilde{V}) &\mapsto (w, \tilde{\neg}(\tilde{V})) \end{aligned}$$

With these definitions, weights are interpreted as threshold values. To illustrate this, consider the following:

- *The choice for the Gödel implication function for the computation of the membership grade of 'true' (T) in the implication operator for conjunction models for example that:*
  - *If the proposition of the criterion is true with a possibility degree $\mu_{\tilde{V}}(T)$ that is larger than or equal to the weight w, then it is assumed that the proposition is completely true, i.e. the modified possibility degree of T becomes 1.*
  - *In the other case it is assumed that the proposition is true with a possibility degree $\mu_{\tilde{V}}(T)$, i.e. the modified possibility degree of T remains $\mu_{\tilde{V}}(T)$.*
- *The choice for the Gödel implication function for the computation of the membership grade of 'false' (F) in the implication operator for conjunction reflects that:*

- *If the proposition of the criterion is false with a possibility degree $\mu_{\tilde{V}}(F)$ that is smaller or equal to the complement $1-w$ of the weight, then it is assumed that the proposition is completely false, i.e. the modified possibility degree of F becomes 0.*
- *In the other case it is assumed that the proposition is false with a possibility degree $\mu_{\tilde{V}}(F)$, i.e. the modified possibility degree of F remains $\mu_{\tilde{V}}(F)$.*

◇

3.4 Frameworks for fuzzy querying

To apply the techniques that are introduced in this chapter in practice, it is necessary to introduce some minimal extensions for traditional database models. These extensions can be easily implemented in a database system. In this section, we briefly handle and describe the (required) extensions for the relational (section 3.4.1) and object oriented database models (section 3.4.2).

3.4.1 Relational databases

Relational databases are constructed in accordance with the prescriptions of the **relational database model**, of which the fundamentals have been presented in 1970 [8, 9]. According to the relational database model, all data are structured in **tables** that have a predefined form. A table is a representation form of the concept **database relation** which on it turn is inspired by the abstract mathematical concept *relation* [26, 18].

Characterizing for relational database relations is that they are all composed of atomic data values, which means that these values are approached by the database model as one single atomic unit and thus are not conceptually further subdivided in components. Each relation R is defined by means of a relation schema

$$R(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$$

that consists of a name R and a finite set of attributes $\{A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n\}$. Each attribute $A_i : T_i$, $1 \leq i \leq n$ on its turn consists of an attribute name A_i which has to be unique within the relation schema, and an associated (atomic) data type which specifies the allowed values for A_i (the values for A_i are restricted to the domain values of T_i). As such, each attribute corresponds with a column in a table. The actual data in a relation are represented by means of a finite set of rows which is called the extent of the relation. The extent of a relation $R(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$ is a finite set that consists of m rows (also called n-tuples)

$$t_i = (A_1 : w_{i,1}, A_2 : w_{i,2}, \ldots, A_n : w_{i,n}), \;\; 1 \leq i \leq m.$$

Each n-tuple t_i consists of n (attribute name, value)-pairs $A_j : w_{i,j}$, $1 \leq j \leq n$ where it must hold that $w_{i,j}$ belongs to the domain of the atomic data type T_j. The values $w_{i,j}$, $1 \leq j \leq n$ together represent (characteristics of) a real world entity. The number of relation attributes is fixed and fully determined by the relation schema. The number of rows, on the other hand, typically changes with the time depending on whether entities are added or removed from the database.

The relational database model prescribes that a relational database schema consists of a finite number of relation schema's. Each relation schema can alternatively be seen as a predicate. For a relation schema $R(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$ this predicate states that 'R models real world entities that are characterized by the attributes with names A_i, $1 \leq i \leq n$ of which the allowed values are determined by their corresponding data type T_i. The rows in the extent of the relation are then propositions that satisfy the predicate. As such, the n-tuple $(A_1 : w_{i,1}, A_2 : w_{i,2}, \ldots, A_n : w_{i,n})$ is interpreted as being a proposition that states that 'there exists a real world entity that is characterized by the value $w_{i,1}$ for attribute A_1, the value $w_{i,2}$ for attribute A_2, ..., and the value $w_{i,n}$ for attribute A_n'. This approach is also extended towards database querying. According to the relational database model, the result of each query is a relation, of which the rows are interpreted as being propositions that satisfy the predicate imposed by the query.

To model relationships between rows of two relations, the concepts 'candidate key' and 'foreign key' have been introduced. A **candidate key** of a relation is an irreducible subset of attributes of that relation, whose attribute values uniquely identify the rows of the relation. This implies that a candidate key is fully characterized by the following two properties:

1. *Unicity*: For each row of the relation, the combination of the values of the attributes that belong to the candidate key must be unique within the relation.
2. *Irreducibility*: It should not be possible to remove an attribute from the candidate key, without violating the unicity property. With other words, no attribute of the candidate key should be redundant.

For each relation in the database schema, at least one candidate key must be defined. One of the candidate keys will be denoted as being the **primary key** of the relation. The relationships between rows of two relations are then modelled by adding a candidate key (in practice usually the primary key) of one relation to the other relation. In this latter relation, the extra added attributes form a so-called **foreign key**. Remark that a foreign key itself is not a candidate key, but only has to contain values that exist as candidate key values in the related relation. The latter constraint is generally known as the referential integrity constraint of relational databases.

In figure 3.4 it is illustrated how the database 'Artworks' of example 3.1 can be modelled in accordance with the prescriptions of the relational database model. The presented database schema consists of three relations 'Painting', 'Artist' and 'Owner'. Because of the assumption that different artists can have the same first name and name, the candidate key of relation 'Artist' is chosen to be

$$\{Name, First\_name, Year\_of\_birth\}.$$

TABLE Painting candidate key: {PID} foreign keys: {Artist} and {Owner}

PID: **STRING**	Name: **STRING**	Artist: **STRING**	Period: **YEAR**	Value: **INTEGER**	Owner: **STRING**
P01	Fishermans house	A04	1882	16.000.000	Boijmans
P02	The ballet course	A02	1872	8.500.000	Louvre
P03	Mona Lisa	A01	1499	75.000.000	Louvre
P04	Afternoon in Ostend	A03	1881	200.000	KMSK

TABLE Artist candidate keys: {AID} and {Name, First_name, Year_of_brith}

AID: **STRING**	Name: **STRING**	First_name: **STRING**	Year_of_birth: **YEAR**	Year_of_death: **YEAR**
A01	Da Vinci	Leonardo	1452	1519
A02	Degas	Edgar	1834	1917
A03	Ensor	James	1860	1949
A04	Monet	Claude	1840	1926

TABLE Owner candidate key: {Name}

Name: **STRING**	Place: **STRING**	Country: **STRING**
Boijmans	Rotterdam	The Netherlands
Louvre	Paris	France
KMSK	Antwerp	Belgium

Fig. 3.4 An example of a relational database schema.

With this choice it is implicitly assumed that there never could be two artists stored in the database who both have the same name, first name and year of birth. An extra so-called surrogate key $\{AID\}$ is added to uniquely identify artists in a more convenient way on the basis of only one attribute. The only candidate key for relation ‘Painting’ is $\{PID\}$, whereas the only candidate key for relation ‘Owner’ is $\{Name\}$. The relationship between a painting and its painter is modelled by the foreign key $\{Artist\}$ in the relation ‘Painting’ that refers to the relation ‘Artist’; the relationship between a painting and its owner is modelled by the foreign key $\{Owner\}$ in the relation ‘Painting’ that refers to the relation ‘Owner’.

The definition and manipulation of a relational database is done by means of the SQL standard language (Structured Query Language) which is based on the operators of the so-called relational algebra[3] that all act upon relations. With that, the relational model is the first database model that has the disposal of a mathematically founded query language.

[3] SQL has also been formalized by means of the so-called relational calculus which is a logical approach.

To come to a framework for flexible querying, the relational database model must be extended. In the remainder of this section we present two possible approaches for such an extension. In the first approach satisfaction degrees are used to model query satisfaction, whereas in the second approach (extended) possibilistic truth values are used.

3.4.1.1 Frameworks based on satisfaction degrees

Structural aspects.

In this approach, it is at least necessary to extend the resulting relations of queries with **one extra attribute** that is suited to model the satisfaction degree of each resulting tuple. This satisfaction degree then denotes to which extent the proposition that corresponds with the tuple satisfies the predicate imposed by the query. Only tuples t of relation r with satisfaction degree $\mu_r(t) > 0$ are included in the relation r. Optionally, the opportunity can be offered to the user to provide a threshold value $0 < \tau \leq 1$ in the query specification. If this is the case, then only the tuples t for which $\mu_r(t) \geq \tau$ holds, are included in the query result. Omitting such an extra attribute would result in an information loss as useful information about query satisfaction would then be discarded.

Another extension is necessary, at least from a theoretical point of view, if we want to keep the *closeness property* of the relational algebra, which states that each operator of the algebra should act upon one or more relations and should result in a new relation such that the result of an operation can be used as argument for another operator and which allows it to build expressions. To guarantee this closeness property, it is recommended to extend each relation R with such an extra attribute. The relation schema of the extended counterpart R^* of a relation

$$R(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$$

is then for example modelled by

$$R^*(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n, degree : real)$$

where *degree* denotes the satisfaction degree of the tuples, which is represented by a real number (of the unit interval $[0,1]$). The constraint that the real number must belong to the unit interval can for example be modelled by adding an extra integrity constraint to the definition of the relation. For regular relations r it must necessarily hold for all tuples t that $\mu_r(t) = 1$.

Operational aspects.

To be supportive for a flexible querying language, the operators of the relational algebra must be adapted in such a way that (the computation of) the satisfaction degree is taking into account. The relational algebra, as originally presented by E.F. Codd [10], consists of eight operators

- *union*
- *intersection*
- *difference*
- *Cartesian product*
- *selection*
- *projection*
- *join*
- *division*

of which the union, difference, Cartesian product, selection and projection form a minimal subset, i.e. the other operators intersection, join and division can be derived from the operators of this minimal subset.

With the understanding that r and r' denote relations and t and t' respectively denote tuples, the operators of the minimal subset can be extended by applying the following computation rules for satisfaction degrees:

- *Union.*
$$\mu_{\text{union}(r,r')}(t) = \max(\mu_r(t), \mu_{r'}(t)).$$
- *Difference.*
$$\mu_{\text{difference}(r,r')}(t) = \min(\mu_r(t), 1 - \mu_{r'}(t)).$$
- *Cartesian product*.
$$\mu_{\text{Cart-prod}(r,r')}(tt') = \min(\mu_r(t), \mu_{r'}(t')).$$
- *Selection.*
$$\mu_{\text{select}(r,c)}(t) = \min(\mu_r(t), \mu_c(t))$$
where c is the (fuzzy) selection condition and $\mu_c(r)$ is the satisfaction degree which results from the evaluation of c with r.
- *Projection.*
$$\mu_{\text{project}(r,V)}(v) = \max_r \mu_r(vw)$$
where V is a subset of the set X of all attributes of r, v is a subtuple consisting of the values for the attributes of V and w is a subtuple consisting of the values for the attributes of $X \setminus V$.

Furthermore, for the intersection it holds that

$$\mu_{\text{intersection}(r,r')}(t) = \min(\mu_r(t), \mu_{r'}(t)).$$

Remark that other extensions are possible and are obtained by considering another (t-norm, t-conorm) couple than $(\min, \max)$.

Moreover, we can also introduce a **support operator** for extended relations which transforms an extended relation into a regular relation that consists of all tuples that belong to the extended relation, i.e.

$$\mu_{\text{support}(r)}(t) = \begin{cases} 1 & \text{iff } \mu_r(t) > 0 \\ 0 & \text{else.} \end{cases}$$

Implementations.

Extensions similar to the ones presented in previous paragraphs have been implemented and used in several fuzzy querying applications. Among the implementations of fuzzy querying engines, we mention the FQUERY package that has been built on top of the Microsoft Access database system [23] and uses a calculus for quantified linguistic propositions that is based on fuzzy logic, the implementation of the SQLf extension of SQL [4], the PRETI-platform which is rather an experimental environment for the exchange of expertise [11, 12] and the FuzzyQueries 2+ software for fuzzy querying of Oracle databases.

3.4.1.2 Frameworks based on with (extended) possibilistic truth values

Structural aspects.

In a approach that is based on a logic with (extended) possibilistic truth values it is at least necessary to extend the resulting relations of queries with **two** —in the case of possibilistic truth values— or **three extra attributes** —in the case of extended possibilistic truth values—. These extra attributes are then used to model the membership grades $\mu_{\tilde{r}_t}(T)$, $\mu_{\tilde{r}_t}(F)$ and $\mu_{\tilde{r}_t}(\perp)$ of the (E)PTV $\tilde{r}_t$ that expresses to which extent it is (un)certain that the predicate of relation r is satisfied with tuple t. Only tuples for which it holds that $\tilde{r}_t$ differs from $\{(F,1)\}$ are included in relation r.

Additionally and for practical reasons, an ordering function *ord* for (E)PTVs can be provided. This allows the user to better interpret results as these can now be ordered on the basis of their associated certainty of query satisfaction.

- A possible ordering function for PTVs is:

$$\begin{aligned} ord :&\tilde{\wp}(I) \to \mathbb{R} \\ &\{(T,\mu_T),(F,\mu_F)\} \mapsto \frac{\mu_T + (1-\mu_F)}{2} \end{aligned}$$

  where $\mathbb{R}$ denotes the set of real numbers.
- A possible ordering function for EPTVs is:

$$ord : \tilde{\wp}(I^*) \to [0,1]$$
$$\{(T,\mu_T),(F,\mu_F),(\perp,\mu_\perp)\} \mapsto \frac{1 + (\mu_T - \mu_F)(1 - \frac{\mu_\perp}{2})}{2}$$

Other ordering functions exist [15]. Also here, the user can be offered the possibility to specify a threshold value $0 < \tau \leq 1$ in the formulation of a query. In this is the case, then only tuples t for which $ord(\tilde{r}_t) \geq \tau$ holds will be included in the result. As an alternative, one can also work with three threshold values τ_T, τ_F and $\tau_\perp$ which all act upon the membership grades from $\tilde{r}_t$. To be included in the query result, a tuple t must satisfy

$$\begin{cases} \mu_{\tilde{r}_t}(T) \geq \tau_T \\ \mu_{\tilde{r}_t}(F) \leq \tau_F \\ \mu_{\tilde{r}_t}(\perp) \leq \tau_\perp \end{cases}$$

To keep the *closeness property* of the relational algebra it is also recommended to extend each relation R with two (or three) extra attributes. The relation schema of the extended counterpart R^* of a relation

$$R(A_1 : T_1, A_2 : T_2, \dots, A_n : T_n)$$

is then for example in the case of three extra attributes modelled by

$$R^*(A_1 : T_1, A_2 : T_2, \dots, A_n : T_n, \mu_T : real, \mu_F : real, \mu_\perp : real)$$

where μ_T, μ_F and $\mu_\perp$ respectively denote the membership grades of the truth values T, F and $\perp$ within the EPTVs of the tuples of R^*. Together the values μ_T, μ_F and $\mu_\perp$ then express to which extent it is (un)certain that the its corresponding tuple satisfies the predicate imposed by the relation schema. Each of the values μ_T, μ_F and $\mu_\perp$ is represented by a real number (of the unit interval $[0,1]$). Like in the approach with satisfaction degrees, the constraint that the real number must belong to the unit interval can for example be modelled by adding an extra integrity constraint to the definition of the relation. For regular relations r it must necessarily hold for all tuples t that $\mu_r(t) = \{(T,1)\}$.

Operational aspects.

To obtain a framework for fuzzy querying the operators of the relational algebra must be adequately adapted in such a way that (the computation of) the associated (E)PTV of each tuple is taking into account.

With the understanding that r and r' are relations and t and t' respectively denote tuples and with the understanding that the notation $e(op)(t)$ is used to denote the evaluation of the operator op for tuple t, the operators union, difference, Cartesian product, selection and projection of the minimal subset of the relational algebra can be extended by applying the following computation rules for (E)PTVs:

- *Union.*
$$e(\text{union}(r,r'))(t) = \tilde{r}_t \mathbin{\tilde{\vee}} \tilde{r'}_t.$$
- *Difference.*
$$e(\text{difference}(r,r'))(t) = \tilde{r}_t \mathbin{\tilde{\wedge}} \tilde{\neg}(\tilde{r'}_t).$$

- *Cartesian product*.
$$e(\text{Cart-prod}(r,r'))(tt') = \tilde{r}_t \,\tilde{\wedge}\, \tilde{r'}_{t'}.$$
- *Selection.*
$$e(\text{select}(r,c))(t) = \tilde{r}_t \,\tilde{\wedge}\, \tilde{c}_t$$
where c is the (fuzzy) selection condition and $\tilde{c}_t$ is the (E)PTV that results from the evaluation of c with t.
- *Projection.*
$$e(\text{project}(r,V))(v) = \underset{t}{\tilde{\vee}}\, \tilde{t}_{vw}$$
where V is a subset of the set X of all attributes of r, v is a subtuple consisting of the values for the attributes of V and w is a subtuple consisting of the values for the attributes of $X \setminus V$.

Furthermore, for the intersection it holds that

$$e(\text{intersection}(r,r'))(t) = \tilde{r}_t \,\tilde{\wedge}\, \tilde{r'}_t.$$

Remark that other extensions are possible and are obtained by considering operators for conjunction ($\tilde{\wedge}$), disjunction ($\tilde{\vee}$) and negation ($\tilde{\neg}$) of (E)PTVs (cf. section 3.2.2.2).

Table 3.1 Intermediate querying result.

PID	*AID*	*Value*	$EPTV_1$	$Year\_of\_death$	$EPTV_2$
P01	A04	3.2M	$\{(T,0.67),(F,1)\}$	1926	$\{(T,0.89),(F,1)\}$
P02	A02	5M	$\{(T,1)\}$	1917	$\{(T,1),(F,0.30)\}$
P03	A01	75M	$\{(T,1)\}$	1519	$\{(F,1)\}$
P04	A03	1.3M	$\{(F,1)\}$	1949	$\{(F,1)\}$

Example 3.7
For the relational database presented in figure 3.4 the following fuzzy query could be considered:

Give the name of the painting and the name of the artist of all very expensive paintings of artists who die at the beginning of the twentieth century, where the condition on the year of death of the artist must have significant larger impact on the query result than the condition on the value of the painting.

This query can be translated to the following algebraic expression:

$$
\begin{aligned}
&project(select(Cart-prod(Painting,Artist),\\
&\quad(Painting.Artist = Artist.AID, weight = 1)\ AND\\
&\quad(Painting.Value\ IS\ \text{‘}very\_expensive', weight = 0.6)\ AND\\
&\quad(Artist.Year\_of\_death\ IS\ \text{‘}beginning\_of\_twentieth\_century', weight = 1)),\\
&\quad\{Painting.Name, Artist.Name\})
\end{aligned}
$$

where ‘very_expensive’ and ‘beginning_of_twentieth_century’ are linguistic terms that are modelled by the fuzzy sets with membership functions

$$
\mu_{very\_expensive}(x) = \begin{cases} 0 & iff\, x < 2M \\ \dfrac{x-2}{3} & iff\, 2M \le x \le 5M \\ 1 & iff\, x > 5M \end{cases}
$$

and

$$
\mu_{beginning\_of\_twentieth\_century}(x) = \begin{cases} 1 & iff\, 1900 \le x \le 1910 \\ 0 & iff\, x < 1900\ or\ x > 1940 \\ \dfrac{1940-x}{30} & iff\, 1910 \le x \le 1940 \end{cases}
$$

The Cartesian product results in a relation with 16 tuples, of which 4 satisfy the join condition $Painting.Artist = Artist.AID$. *We can continue working with these 4 tuples because of the facts that the associated weight of the join condition is 1 and that there are only conjunction operators in the composed query condition. When the logical framework based on EPTVs is used, the evaluation of the simple fuzzy conditions*

$$Painting.Value\ IS\ \text{‘}very\_expensive'$$

and

$$Artist.Year\_of\_death\ IS\ \text{‘}beginning\_of\_twentieth\_century'$$

results respectively in the EPTVs $EPTV_1$ *en* $EPTV_2$ *that are presented in table 3.1. Using the extended conjunction operator* $\wedge^{w}{}_{prob}$ *from example 3.6 we obtain with the given weights* $w_1 = 0.6$ *and* $w_2 = 1$ *the intermediate aggregated querying result that is presented in table 3.2. For each of the intermediate resulting tuples, the associated aggregated weight is* $w = 1$. *The final querying result, which is obtained after*

Table 3.2 Intermediate aggregated querying result.

SID AID $EPTV_1 \wedge^{w}{}_{prob} EPTV_2$
P01 A04 $\{(T,1),(F,1)\}\ \tilde{\wedge}^{w}{}_{prob}\ \{(T,0.89),(F,1)\} = \{(T,0.89),(F,1)\}$
P02 A02 $\{(T,1)\}\ \tilde{\wedge}^{w}{}_{prob}\ \{(T,1),(F,0.30)\} = \{(T,1),(F,0.30)\}$
P03 A01 $\{(T,1)\}\ \tilde{\wedge}^{w}{}_{prob}\ \{(F,1)\} = \{(F,1)\}$
P04 A03 $\{(F,1)\}\ \tilde{\wedge}^{w}{}_{prob}\ \{(F,1)\} = \{(F,1)\}$

projection, is given in table 3.3. ◇

Table 3.3 Final querying result.

$Painting.Name$	$Artist.Name$	μ_T	μ_F	$\mu_\perp$
'Fishermans house'	'Monet'	0.89	1	0
'The ballet course'	'Degas'	1	0.30	0

Implementations.

The extension presented above has been implemented in a DBMS independent prototype of a fuzzy querying engine for relational databases [16].

3.4.2 Object oriented approaches

Along with the success of the object oriented paradigm and object oriented programming languages like C++, Smalltalk and Java came the need for advanced database facilities to manage *complex objects*, with support for typical object oriented facilities like, among others, *object identity*, *encapsulation of structure and/or behaviour*, *inheritance* and *type hierarchies*, *polymorphism* and *operator overloading*. To cater for this extra need, several new database models have been developed. These models can grosso modo be subdivided in two categories:

1. Database models that fully support the object oriented paradigm and are therefore called **object oriented database models**. An (unsuccessful) attempt to standardize these approaches resulted in the so-called ODMG model [6].
2. Database models that support, and are in fact extensions of, the relational database model and are therefore called **object relational database models**. These approaches have less object oriented facilities than their object oriented counterparts, but are standardized in the SQL99 and more recent SQL3 standards.

The **object oriented database models** are proposed in the beginning of the 90's and are all built around pure object oriented concepts. By specifying data structures in (almost) the same way in both the programming language and the database model, data can almost seamlessly be communicated between (application) programs and databases. Common to all object oriented database models is that they all support a notion of **objects** which all have a unique identity and are defined by means of one or more data types or object prototypes [24] wherein the structure and associated behaviour of the object are defined. The structure of an object is typically defined by means of attributes and (binary) relationships which allow to relate the object with other objects; the behaviour of an object is determined by the operators that act upon the object.

Due to the large diversity and differences of concept definitions in object oriented data models and programming languages, there exists a large variety of possible definitions and interpretations for the concept object. This has been resulted in a lack of uniformity within the object oriented programming paradigm and a large diversity of object oriented database models. This problem has already been identified and mentioned in 1989 [25, 1] and is until now one of the most important obstructions for the standardization and commercial break through of *the* object oriented database model. Although not commonly accepted, is it worth to mention the ODMG object model (Object Database Management Group) [6] as a proposal for such a standard.

Figure 3.5 illustrates the modelling of the database for artworks of example 3.1 in accordance with the prescriptions of the ODMG object model. Objects are spec-

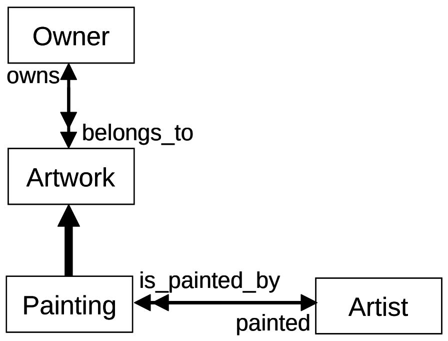

Fig. 3.5 An example of a graphical representation of an ODMG database schema.

ified by means of so-called **classes** which help to define the object definitions and are represented in the figure by rectangles. In the figure classes are provided for the specification of painting objects (‘Painting’), objects representing owners (‘Owner’) and objects representing artists (‘Artist’). For the sake of illustration we have also provided a more general class ‘Artwork’ which helps to determine, by means of a mechanism called ‘inheritance’, the specification of a painting object (this is depicted in the figure by the fat arrow that is drawn from the more specific class ‘Painting’ to the more general class ‘Artwork’). Association relations between objects are specified by means of binary relationships that are defined between the classes of these objects and are depicted in the figure by thin labelled two-sided arrows. To specify the semantics of the binary relationship, a cardinality constraint is associated with each of the participating classes (a cardinality 1 is depicted by a single arrow, whereas a cardinality *many* is depicted by a double arrow). Further, each thin arrow is labelled with the names of the relation as specified in each of the participating classes. For example, the thin double-sided arrow between ‘Owner’ and ‘Artwork’ represents that each artwork object belongs to (‘belongs_to’) one owner and that each owner owns (‘owns’) one or more artwork objects. Analogously, the thin double-sided arrow between ‘Painting’ and ‘Artist’ represents that each painting is painted by (‘is_painted_by’) one artist and that each artist has painted (‘has_painted’) one or more paintings.

Alternative, less rudimentary graphical modelling conventions can be used. In this context UML [19, 2] is worth mentioning. UML allows it to represent both the structural and the behavioural characteristics of classes. However, according to some database experts UML is too stringent and too closely connected to programming languages and is therefore considered as being less suited for database modelling purposes which should be database model independent. A good alternative, which is database model independent, but does not support the modelling of behaviour, is the Enhanced Entity Relationship modelling technique [7, 27, 22].

The **object relational database models** are all extensions of the relational database model. With the **SQL3 standard** the SQL standard has been evolved towards a standard for object relational database models. SQL3 provides, among others, in facilities

- to construct structured user defined data types that can be used for the construction of complex relations of which the attributes no longer need to be atomic;
- to associate operators with relations;
- to reuse specification of existing relations for the construction of a new relation (inheritance);
- to support tuple-identity which allows to directly navigate from one tuple to its associated tuples without the need of resource consuming join operations, and
- to support unstructured data types for the management of multimedia and textual information.

The difference with the object oriented database models is that the standardized object relational model is based on the relational database model instead of on a ‘pure’ notion of classes and objects. Thanks to their underlying standardization, their underlying commonly used relational structure, their relative simplicity and their mathematically supported behaviour, object relational database systems can continue profiting from the commercial success of the relational database systems. Even stronger, they completely push out the object oriented database systems from the market. The most important commercial relational database systems are nowadays almost all evolved to an object relational system.

3.4.2.1 Structural aspects

The structural extensions that are required for flexible querying of object oriented and object relational databases are analogous to these of relational databases: depending on the chosen framework each database object must be associated with a satisfaction degree or an (E)PTV. From a theoretical point of view, this can be done without the use of extra attributes: the satisfaction degree or (E)PTV can be seen as an extra feature of the persistent object (or tuple) and directly be associated with the object identifier or tuple identifier (in case the latter does not exist, the primary key can be used). These associated values, either satisfaction degrees or (E)PTVS, must be visible for the users. In practice, one can always provide extra attributes in some system owned data types whose characteristics have to be inherited by all object types that are part of a database schema.

3.4.2.2 Operational aspects

With respect to the operational extensions, two aspects must be considered:

1. Object relational databases use the SQL3 querying language. The basic operators of this language are the same as in SQL and can thus be extended analogously as with relational databases. Object oriented databases that are compliant with the ODMG model are queried using the OQL querying language. This language is also based on SQL and can thus also be extended analogously as with relational databases.
2. Object oriented and object relational database models allow the user to provide user-defined operators. These operators can then be used for the querying and manipulation of objects (or tuples) and can in fact be integrated in queries. The facility of user-defined operators can skilfully be used for the implementation of a flexible querying engine.

3.4.2.3 Implementation

As far as we know there are no implementations of flexible querying engines that are intended to act on exclusively on regular object oriented databases. However, there exist implementations of flexible querying engines for ‘fuzzy’ object oriented databases. These implementations are described in chapters 4 and 5.

References

1. P.A. Bernstein, U. Dayal, D.J. DeWitt, D. Gawlick, J. Gray, M. Jarke, B.G. Lindsay, P.C. Lockemann, D. Maier, E.J. Neuhold, A. Reuter, L.A. Rowe, H.J. Schek, J.W. Schmidt, M. Schrefl, and M. Stonebraker, “Future Directions in DBMS Research (The Laguna Beach report)”, *ACM SIGMOD Record* **18** 1 (1989) 17–26.
2. G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide* (Addison-Wesley, Reading, USA, 1999).
3. P. Bosc, and O. Pivert, “Some approaches for relational databases flexible querying”, *International Journal on Intelligent Information Systems* **1** (1992) 323–354.
4. P. Bosc, and O. Pivert, “SQLf: A Relational Database Language for Fuzzy Querying”, *IEEE Transactions on Fuzzy Systems* **3** (1995) 1–17.
5. P. Bosc, D. Kraft, and F. Petry, “Fuzzy sets in database and information systems: status and opportunities”, *Fuzzy Sets and Systems* **153** 3 (2005) 418–426.
6. R.G.G. Cattell, and D.K. Barry (eds.), *The Object Data Standard: ODMG 3.0* (Morgan Kaufmann Publishers Inc., San Francisco, USA, 2000).
7. P.P. Chen, “The Entity-Relationship Model — Toward a Unified View of Data”, *ACM Transactions on Database Systems* **1** 1 (1976).
8. E.F. Codd, “A Relational Model of Data for Large Shared Data Banks”, *Communications of the ACM* **13** 6 (1970). Republished in *Communications of the ACM* **26** 1 (1983).
9. E.F. Codd, “Further Normalization of the Database Relational Model”, in: *Database Systems*, Prentice-Hall, New Jersey, USA (1971) 65–98.
10. E.F. Codd, “Relational Completeness of Data Base Sublanguages”, in: *Data Base Systems, Courant Computer Science Symposia Series 6*, R.J. Rustin (ed.) Prentice-Hall, Englewood Cliffs, USA (1972).
11. M. De Calmès, D. Dubois, E. Hüllermeier, H. Prade, and F. Sedes, “A fuzzy set approach to flexible case-based querying: methodology and experimentation”, in: *Proc. of the 8th International Conference, Principles of Knowledge Representation and Reasoning (KR2002)* (Toulouse, France, 2002) 449–458.
12. M. De Calmès, D. Dubois, E. Hüllermeier, H. Prade, and F. Sedes, “Case-based querying and prediction: a fuzzy set approach”, in: *Proc. of 2002 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE’02)* (Honolulu, Hawaii, 2002) 735–739.
13. G. De Tré, R. De Caluwe, J. Verstraete, and A. Hallez, “Conjunctive Aggregation of Extended Possibilistic Truth Values and Flexible Database Querying”, *Lecture Notes in Artificial Intelligence* **2522** (2002) 344–355.
14. G. De Tré, A. Hallez, J. Verstraete, and A. Verkeyn, “Beyond Conjunctive Aggregation of Possibilistic Truth Values in Database Systems”, in: *Proceedings of the Eurofuse 2002 Workshop on Information Systems* (Varenna, Italy, 2002) 137–142.

15. G. De Tré, T. Matthé, K. Tourné, and B. Callens, “Ranking the Possible Alternatives in Flexible Querying: An Extended Possibilistic Approach”, *Lecture Notes in Computer Science* **2869** (2003) 204–211.
16. G. De Tré, R. De Caluwe, K. Tourné, and T. Matthé, “Theoretical considerations ensuing from experiments with flexible querying”, in: *Proceedings of the IFSA 2003 World Congress* (Istanbul, Turkey, 2003) 388–391.
17. D. Dubois, H. Fargier, and H. Prade, “Beyond min aggregation in multicriteria decision: (ordered) weighted min, discri-min and leximin”, in: *The ordered weighted averaging operators: Theory and applications.*, R.R. Yager, and J. Kacprzyk (eds.) (Kluwer Academic Publishers, Boston, USA, 1997) 181–192.
18. R. Elmasri, and S.B. Navathe, *Fundamentals of Database Systems*, fourth edition (Pearson Education, Boston, USA, 2004).
19. M. Fowler, and K. Scott, *UML Distilled: Applying the Standard Object Modeling Language* (Addison Wesley Longman, Inc., Reading, USA, 1997).
20. T. Gaasterland, P. Godfrey, and J. Minker, “An overview of cooporative answering”, *Journal of Intelligent Information Systems* **1** (1992) 123–157.
21. J. Galindo, J.M. Medina, O. Pons, and J.C. Cubero, “A Server for Fuzzy SQL Queries”, in: *Flexible Querying and Answering Systems*, T. Andreasen, H. Christiansen en H.L. Larsen (eds.) (Kluwer Academic Publishers, Dodrecht, The Netherlands, 1998) 164–174.
22. M. Gogola, and U. Hohenstein, “Towards a Semantic View of an Extended Entity-Relationship model”, *ACM Transactions on Database Systems* **16** 3 (1991).
23. J. Kacprzyk, and S. Zadrozny, “The paradigm of computing with words in intelligent database querying”, in: *Computing with Words in Information/Intelligent Systems. Part 2. Applications.* L.A. Zadeh, and J. Kacprzyk (eds.) (Physica-Verlag, Heidelberg, Germany, 1999) 382–398.
24. H. Lieberman, “Using prototypical objects to implement shared behavior in object-oriented systems”, *ACM SIGPLAN Notices* **21** 11 (1986) 214–223.
25. D. Maier, “Why isn’t there an Object-Oriented Data Model?”, in: *Proceedings of the IFIP Information Processing ’89 conferentie*, Elsevier Science Publishers B.V., Amsterdam, The Netherlands (1989) 793–798.
26. J. Paredaens, P. De Bra, M. Gyssens, and D. Van Gucht, *The Structure of the Relational Database Model* (Springer-Verlag, Heidelberg, Germany, 1989).
27. T. Teorey, D. Yang, and J. Fry, “A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship model”, *ACM Computing Surveys* **18** 2 (1986).
28. M. Umano, and S. Fukami, “Fuzzy Relational Algebra for Possibility-Distribution-Fuzzy Relational Model of Fuzzy Data”, *Journal of Intelligent Information Systems* **3** (1994) 7–27.
29. S. Zadrozny, and J. Kacprzyk, “FQUERY for Access: towards human consistent querying user interface”, in: *Proceedings of the 1996 ACM Symposium on Applied Computing (SAC)* (Philadelphia, USA, 1996) 532–536.

Chapter 4
Fuzzy databases

By providing also at the level of database modelling facilities to model imperfect data as adequate as possible, even more flexibility in database systems could be obtained. Such an approach results in databases that are semantically richer. If the modelling of imperfections is based on *fuzzy set theory* [60] one speaks about a **'fuzzy' database**.

Radecki was one of the firsts to recognize the power of fuzzy set theory with respect to information management [47]. After Radecki, several other researchers have proposed 'fuzzy' database models. As such, there exist extensions and generalizations of the relational database model, the (Enhanced) Entity Relationship model, object relational models and object oriented models. For an overview we want to refer to [9, 44, 19, 59, 8, 38].

This chapter deals in more detail with the modelling of imperfect data in databases and is subdivided in two sections. In the first section 4.1 we handle the data modelling aspects. Hereby we subsequently describe techniques for the modelling of imprecise, uncertain, incomplete, and inconsistent information. In the second section 4.2 we present some fuzzy database models.

4.1 Data modelling

4.1.1 Dealing with imprecise and fuzzy information in databases

Possibility theory [62, 25] is the most commonly used methodology for the modelling of imprecise and fuzzy data in databases. Another approach is to work with similarity relations [10]. More recently, some other alternatives have been proposed, making use of generalizations of fuzzy sets [22].

In each of these approaches the data types in the field definitions of the record types are *extended* or *generalized* in such a way that the domain values could also be fuzzy sets.

In an **extension**, some extra data types like

$$FUZZYINTEGER, FUZZYREAL \text{ and } FUZZYSTRING$$

will be added to the database model. The domain of such an extra data type is defined to be the fuzzy powerset of the domain of the corresponding regular data type. As such, the domain

$$dom_{FUZZYINTEGER} = \tilde{\wp}(dom_{INTEGER})$$

will for example be the set of all fuzzy sets that could be defined over the domain of the data type '*INTEGER*'.

In a **generalization**, the existing regular data types are generalized in such a way that their domains also contain fuzzy sets. The original domain values x are then considered to be the same as the fuzzy set $\{(x,1)\}$.

Both approaches have their own advantages and disadvantages. With an extension we have the theoretical problem that we can model perfect information in two ways: via the regular data types and via the extended data types. Moreover, the database designer must decide in advance whether a record field can contain fuzzy values or whether it is restricted to perfect values. The advantages of an extension are that it is easier to implement and that it could be integrated in an easier way in existing databases (because almost no database conversion is needed). Generalizations are a better solution from a theoretical point of view, but are more difficult to implement.

4.1.1.1 Possibilistic approaches

In a possibilistic approach, the membership grades of the fuzzy sets are interpreted as degrees of uncertainty. As such, the fuzzy set can be seen as a possibility distribution that is associated with the record field A and therefore is denoted as

$$\pi_A.$$

Because of imprecision, the exact value of the record field in not known with certainty. The possibility that $A = x$ is $\pi_A(x)$, $x \in dom_t$, with dom_t the domain of the data type of the record field.

For a single-valued field, the possible values $x \in dom_t$ are represented by a possibility distribution that is defined on dom_t, which necessarily must consist of single-valued elements. For a multi-valued field, the domain dom_t consists of multi-valued elements (collections). Each collection x for which $\pi_A(x)$ differs from 0 then a candidate value for the field A, where the associated membership grade $\pi_A(x)$ denotes the possibility, i.e., the degree of (un)certainty, that A has the collection x as value.

Example 4.1
The value of the record field 'Value' of the record type 'Painting' of the example database of example 3.1 could for a given record be vaguely described as 'very_valuable' and be defined by the possibility distribution

$$\pi_{Value}(x) = \mu_{very\_valuable}(x) = \begin{cases} 0 & \text{iff } x < 10M \\ \dfrac{x-10}{10} & \text{iff } 10M \leq x \leq 20M \\ 1 & \text{iff } x > 20M \end{cases}$$

◇

4.1.1.2 Similarity-based approaches

In the similarity-based approaches [10] the basis domains dom_t of the data types t of the fields of the record types are extended with a similarity relation S (cf. subsection 3.2.1.3) and generalized to power sets $\wp(dom_t)$. For some domains the similarity relation will be the identity relation I, which is a special kind of similarity relation and is defined by:

$$\begin{aligned} I : U \times U &\to [0,1] \\ (x,y) &\mapsto 1 \text{ iff } x = y \\ (x,y) &\mapsto 0 \text{ else} \end{aligned}$$

Due to the generalization, all field values of a record field with data type t are sets that are elements of $\wp(dom_t)$. The similarity relation defines the degrees of correspondence, which denote to what extent the elements of the field value could be used instead of each other. Several valid interpretations are associated with a record

$$[c_1 : V_1; c_2 : V_2; \ldots; c_n : V_n], n \in \mathbb{N} \setminus \{0\}, V_i \subseteq U, 1 \leq i \leq n$$

An interpretation

$$\alpha = [c_1 : a_1, c_2 : a_2, \ldots, c_n : a_n]$$

is obtained by choosing an element a_i in each of the sets V_i and by considering these elements together. To be valid, an interpretation must moreover satisfy the underlying semantics of the record type. As such, the interpretation

$$[\text{Brussels}, \text{France}]$$

of the record

$$[\{\text{Brussels}, \text{Warsaw}, \text{Paris}\}, \{\text{Belgium}, \text{Poland}, \text{France}\}]$$

of the record type

$$Capital(City : CHAR(30); Country : CHAR(30))$$

is invalid.

4.1.1.3 Other approaches

Different generalizations of the concept of ‘fuzzy set’ have been presented in literature for several reasons. Some of these generalizations have in essence been developed to deal simultaneously with both positive and negative knowledge (bipolarity), which in some situations can be beneficial because it allows to prevent information loss.

Three such generalizations, which could be beneficial in the context of fuzzy databases, are the so called *‘interval-valued’ fuzzy sets* (IVFS), the *‘intuitionistic’ fuzzy sets* (IFS) and the *‘two-fold’ fuzzy sets* (TFS). Each of these generalizations allow to model the semantics of information in a more natural way.

‘Interval-valued’ fuzzy sets.

An IVFS

$$F = \{< u, \mu_F^l(u), \mu_F^u(u) > | u \in U\}$$

over a universe U is defined by two functions

$$\mu_F^l, \mu_F^u : U \to [0,1]$$

such that

$$0 \leq \mu_F^l(u) \leq \mu_F^u(u) \leq 1, \forall\, u \in U.$$

For each $u \in U$, the values $\mu_F^l(u)$ and $\mu_F^u(u)$ respectively denote a lower and upper bound for the membership grade of u in F. It follows clearly from the special case where $\mu_F^l = \mu_F^u$ that ‘interval-valued’ fuzzy sets are generalizations of regular fuzzy sets.

If an IVFS is used for the modelling of imprecise or vague data in a fuzzy database, then the lower and upper bounds can be assigned a possibilistic interpretation such that the IVFS becomes a representation of a generalized possibility distribution [40]. An ‘interval-valued’ possibility distribution (IVPD) can be associated with a record field A, in which case it could be denoted as

$$(\pi_A^l, \pi_A^u)$$

and could be characterized by the functions π_A^l and π_A^u, such that

$$0 \leq \pi_A^l(u) \leq \pi_A^u(u) \leq 1, \forall\, u \in U.$$

Furthermore, $\pi_A^l(u)$ is interpreted as a lower bound for the possibility that $A = u$ and $\pi_A^u(u)$ is interpreted as an upper bound for the possibility that $A = u$. In this way, a IVPD makes it possible to model the imprecision of the data in a database as well as the uncertainty that is inherently connected with this modelling.

An IVFS can also be used within the fuzzy querying of regular databases in cases where the membership grades (of linguistic terms) could not be exactly assigned.

As an example, consider the case where a user wants to specify that the age 30 is compatible with the linguistic term ‘young’ with a degree between 0.4 and 0.6.

‘Intuitionistic’ fuzzy sets.

An IFS [2]

$$F = \{< u, \mu_F(u), \nu_F(u) > | u \in U\}$$

over a universe U is defined by two functions

$$\mu_F, \nu_F : U \rightarrow [0,1]$$

such that

$$0 \leq \mu_F(u) + \nu_F(u) \leq 1, \forall\, u \in U.$$

For each $u \in U$, the values $\mu_F(u)$ and $\nu_F(u)$ respectively represent the grade of membership and the grade of non-membership of u in F. Considering the special case where $\nu_F = 1 - \mu_F$, it follows clearly that ‘intuitionistic’ fuzzy sets are generalizations of regular fuzzy sets. Even more, by taking $\mu_F^l = \mu_F$ and $\mu_F^u = 1 - \nu_F$ it follows that an IFS can formally be dealt with as an IVFS. A further study on this correspondence is outside the scope of this book. In what follows, we are only interested in the interpretation and usability of both generalizations of fuzzy sets within the context of fuzzy databases.

If an IFS is used for the modelling of imprecise or vague data in a fuzzy database, then a possibilistic interpretation can be assigned to it such that also in this case a kind of generalized possibility distribution is obtained. An ‘intuitionistic’ possibility distribution (IPD) can be associated with a record field A, in which case it could be denoted as

$$(\pi_{\mu_A}, \pi_{\nu_A})$$

and could be characterized by the functions π_{μ_A} and π_{ν_A}, such that

$$\pi_{\mu_A}(u) + \pi_{\nu_A}(u) \leq 1, \forall\, u \in U.$$

Furthermore, it is assumed that $\pi_{\mu_A}(u)$ defines the degree that it is possible that $A = u$ and $\pi_{\nu_A}(u)$ defines the degree that it is impossible that $A = u$. In the case of a regular possibility distribution π_A this degree of impossibility equals $1 - \pi_A(u)$ and is thus completely determined by $\pi_A(u)$. The notion of impossibility is fairly intuitive and also fits formally in the regular possibilistic context because $\nu_F(u)$ in an IFS corresponds with $1 - \mu_F(u)$ in a regular fuzzy set. Consequently, $1 - \pi_{\mu_A}(u)$ and $\pi_{\nu_A}(u)$ could be interpreted as the necessity that $A \neq u$.

Example 4.2
The idea of using ‘intuitionistic’ fuzzy sets and their generalized possibility distributions for the modelling of data in fuzzy databases can be illustrated as follows: consider a database record type ‘Crime’ which is used among others to keep in-

formation about possible perpetrators. For the sake of simplicity it is assumed that each crime is committed by exactly one perpetrator, which is in a list of suspects. To model this, the record type has a field 'Perpetrator'. The values of this field could be modelled by means of an IPD which is completely determined by an IFS of suspects. Such an IFS, F, can for example be constructed by means of the following procedure. A group of experts studies the crimes and characteristics of the suspects under consideration. After that, each expert votes for each of the considered suspects. Voting results in:

- *'yes' if the expert is convinced that the suspect is a potential perpetrator of the crime;*
- *'no' if the expert is convinced that the suspect is not a potential perpetrator of the crime;*
- *if the expert cannot decide between 'yes' and 'no', he or she can abstain.*

After the voting, the proportion of 'yes' and 'no' votes can respectively be used to determine the membership functions μ_F and ν_F of the IFS F. Finally, the corresponding IPD distribution denotes for each suspect how possible (π_{μ_F}) and how impossible (π_{ν_F}) it is that he or she has committed the crime. $\diamond$

An IFS can also be used in a more intuitive way in the fuzzy querying of regular databases to denote which field values are preferable in the search —given via the membership function μ_{π_A}— and which field values should be avoided —given via the membership function ν_{π_A}.

Twofold fuzzy sets.

A TFS [24]

$$F = (\{< u, \mu_{F^P}(u) > | u \in U\}, \{< u, \mu_{F^S}(u) > | u \in U\})$$

over a universe U is defined as a couple of fuzzy sets

$$F^P = \{< u, \mu_{F^P}(u) > | u \in U\} \text{ and } F^S = \{< u, \mu_{F^S}(u) > | u \in U\}$$

over U, such that

$$\text{supp}(F^P) \subseteq \text{core}(F^S).$$

Consequently, it holds that

$$0 \leq \mu_{F^P}(u) \leq \mu_{F^S}(u) \leq 1, \forall\, u \in U.$$

From the special case $\mu_F^l = \mu_{F^P}$ and $\mu_F^u = \mu_{F^S}$ and the special case $\mu_F = \mu_{F^P}$ and $\nu_F = 1 - \mu_{F^S}$, it follows clearly that a TFS can formally be dealt with as a special case of respectively an IVFS or an IFS set. In what follows, we are only interested in the interpretations and usability of twofold fuzzy sets in the context of fuzzy databases.

A TFS can be used in the querying of regular databases to denote which field values are satisfactory (allowed, acceptable, not rejected) and which of these values are really preferred, cf. [27]. The membership grades of the preferred values correspond with μ_{F^P}, the membership grades of the satisfactory values with μ_{F^S}.

Due to the correspondence between on the one hand twofold fuzzy sets and on the other hand both interval-valued fuzzy sets as well as intuitionistic fuzzy sets, twofold fuzzy sets could also be used to model imprecise or vague data in databases. In such cases, a TFS can be assigned a possibilistic interpretation, by which another kind of generalized possibility distribution is obtained. A twofold possibility distribution (TPD) can then be associated with a record field A, in which case it could be denoted as

$$(\pi_A^P, \pi_A^S)$$

and could be characterized by the functions π_A^P and π_A^S, such that

$$0 \leq \pi_A^P(u) \leq \pi_A^S(u) \leq 1, \forall\, u \in U.$$

Herewith, it is assumed that $\pi_A^P(u)$ defines the possibility that u is a preferred value for A and that $\pi_A^S(u)$ defines the possibility that u is a satisfactory value for A. In this way, π_A^P puts a 'hard' constraint on the possible values for A, whereas π_A^S puts a 'soft' constraint on these values.

4.1.2 Dealing with uncertain information in databases

To deal with uncertain information in databases we can use *possibility theory* [62, 25] in a completely analogous way as with the handling of imprecise and vague information. On the other hand, if one has more control on the data that has to be modelled, as explained in section 2.2, then one can also use *probability theory* [51, 28]. In both approaches, uncertainty is modelled by means of an uncertainty distribution, which associates with each candidate value a value that expresses to what extent it is certain (or uncertain) that the candidate value would be the effective value. The way how these associated values are interpreted and must be processed is completely defined by the used theory. Beside probability and possibility theory, there exist other theories for the modelling of uncertain information. An overview can be find in the work of Peter Walley [56, 57].

In the simplest approaches based on **probability theory**, a *probability* is associated with each database record (a tuple if the relational database model is used [58, 45] or a persistent object of an object type if an object oriented database model is used [33]). These probabilities form a probability distribution which in turn is used in the query processing to model to which extent it is certain (or uncertain) that the record represents a correct answer to the query. In case of the relational database model, probability distributions could also be used to model uncertain attribute values [3, 35]. The same holds for object oriented database models where there exist also proposals to model uncertain inheritance with probability distributions [33].

In fuzzy databases, uncertainty is in most cases modelled by means of **possibility theory**. This is in essence due to the more conservative character of the theory, which results, among others, in a less stringent normalization condition (cf. section 2.2), that is better suited in practice. When using possibility theory, uncertainty is modelled by means of possibility distributions, which in most cases could be derived from a fuzzy set with discrete membership function. Possibility distributions can be associated with attributes in order to model uncertain attribute values of database tuples or database objects [46]. Furthermore, degrees of uncertainty could be associated with database tuples [54] and database objects [53]. In such a case, the associated degree of uncertainty denotes to which extent it is certain or uncertain that the tuple (resp. object) belongs to the extent of the relation (resp. object type). Possibilistic uncertainty can also be considered in object oriented inheritance [53].

Because of the fact that imprecision and vagueness are both orthogonal with respect to uncertainty (cf. section 1.1.2), it could happen that a possibility distribution for the modelling of uncertainty is defined on a universe of possibility distributions for the modelling of imprecision or vagueness. In such a case, the possibility distribution is derived from a so-called **level-2 fuzzy set** [21].

A level-2 fuzzy set $\tilde{\tilde{V}}$ over a universe U [61, 30] can informally be described as a fuzzy set of which the elements are regular fuzzy sets that are all defined over U. This can formally be defined as follows:

Definition 4.1 (Level-2 fuzzy set) *A level-2 fuzzy set $\tilde{\tilde{V}}$ over a universe U is defined by*

$$\tilde{\tilde{V}} = \{(\tilde{V}, \mu_{\tilde{\tilde{V}}}(\tilde{V})) | \forall\, \tilde{V} \in \tilde{\wp}(U) : \mu_{\tilde{\tilde{V}}}(\tilde{V}) > 0\}$$

where each regular fuzzy set $\tilde{V}$ is defined by

$$\tilde{V} = \{(x, \mu_{\tilde{V}}(x)) | \forall\, x \in U : \mu_{\tilde{V}} > 0\}$$

□

Example 4.3
Suppose that it is only known that the value of a painting is either 'most probably about 2M*', or 'less probably* 3.2M*'. This information can for example be modelled by a possibility distribution which is derived from a level-2 fuzzy set*

$$\{(about\_2M, 1), (\{(3.2M, 1)\}, 0.7)\}$$

where $\{(3.2M, 1)\}$ is a possibility distribution and 'about_2M' is a linguistic term, that is modelled by the possibility distribution

$$\pi_{Value}(x) = \mu_{about\_2M}(x) = \begin{cases} 0 & \text{iff } x < 1.8M \text{ or } x > 2.2M \\ \dfrac{x - 1.8}{0.2} & \text{iff } 1.8M \le x \le 2M \\ \dfrac{2.2 - x}{0.2} & \text{iff } 2M \le x \le 2.2M \end{cases}$$

◇

By applying the extension principle (cf. section 2.1.4), the operators for regular fuzzy sets can be generalized to operators for level-2 fuzzy sets [21]. More details on this are beyond the scope of this book.

4.1.3 Dealing with incomplete information in databases

In regular databases, missing information is in most cases modelled by means of a pseudo description called **null**, which denotes that the actual database value is missing [13, 55, 52, 4, 50, 31, 63, 14, 32]. As soon as null values are allowed in a database, it is also necessary to define their impact on the database manipulation and the database querying [39]. In Codd's approach, the relational calculus is extended using an underlying three-valued logic [13, 14] to formally define the semantics of null values in relational databases. This approach is being criticized a lot due to the fact that the law of excluded middle does not hold in a three valued logic [15, 16, 17].

As an alternative, Date proposed to avoid using null values by defining an appropriate, so-called **default** value for each record field which could contain missing information. The default value must be an element of the domain of the data type of the record field. Whenever data is missing in those fields, this data will be approximately represented by the default value [15, 18]. Default values have as disadvantage that they can give a misrepresented view of reality, what especially can cause problems in statistical querying.

Both the approach with null values and the approach with default values have been generalized for fuzzy databases. In the framework of possibility theory the problems with null values can be solved to a large extent. In what follows, we only describe such a generalization of a null value approach. For a description of a generalization of a default value approach we refer interested readers to [1, 42, 43].

In order to assign correct semantics to null values it is important to make a distinction between two main sources of missing information in databases (see also section 1.1.3). As originally presented by Codd [14] missing information can occur because:

- Data are *unknown* to the users. In such a case, the data exist but they cannot be entered in the database because they are unavailable at the moment they has to be entered.
- Data are missing because they are related to a property that does not apply for the database record under consideration. In such a case one speaks about *undefined data*.

To illustrate these two cases, we consider a database that contains information about birds. For each bird type the fly speed is among other registered in the database. The first case, unknown data, occurs for example in a situation where it is known that the birds of that type can fly, but extra observations are still necessary to know the fly speed. The second case, undefined data, occurs when it is known that birds of the type under consideration cannot fly, as for example is the case with penguins (fly speed is not applicable for penguins).

The truth values in Codd's original three-valued logic are respectively 'true' (T), 'false' (F) and 'unknown' ($\perp_{null}$). Hereby, the truth value 'unknown' is used to model logical expressions involving either unknown or undefined data. Furthermore, this logic is a strong Kleene logic [48] (cf. section 2.4.1), which means that the computation rules for negation, conjunction and disjunction of section 2.4.1 hold.

The law of excluded middle does not hold in a strong Kleene logic. This can be seen in

$$\perp_{null} \wedge \neg(\perp_{null}) = \perp_{null} \neq F$$

and

$$\perp_{null} \vee \neg(\perp_{null}) = \perp_{null} \neq T.$$

Furthermore, 'unknown' denotes the *uncertainty* one has about the fact whether the proposition is 'true' or 'false'. This completely differs from the underlying principle of many-valued logics that states that different grades of truth are considered: grades of uncertainty and grades of truth are from the semantic point of view completely different concepts [26]. So, using an extra truth value to model 'unknown' does not make sense from such a point of view. These observations explain partly the motivations behind the criticism of Date on the use of null values [15, 16, 17]. In the modelling approach described in the next paragraphs, unknown information – more specifically uncertainty, due to unavailibility, about the actual values of record fields – is modelled by means of possibility theory, which is in fact meant for the modelling of uncertainty [62, 25].

4.1.3.1 The modelling of unknown information

In fuzzy databases, 'unknown' can be modelled by means of a normalized possibility distribution which is characterized by a fuzzy set with a membership function that takes the value 1 for all regular domain values. An extra null value for the modelling of unknown data is meaningless and thus not required in fuzzy databases.

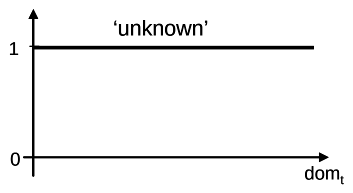

Fig. 4.1 Value unknown.

Example 4.4
If the value of a given painting is unknown, this can be modelled by a possibility distribution which is characterized by a fuzzy set unknown with membership function

$$\begin{aligned}\mu_{unknown} : dom_{value} &\to [0,1]\\ x &\mapsto 1\end{aligned}$$

This is illustrated in figure 4.1. $\diamond$

4.1.3.2 The modelling of undefined information

A specific null value is still required for the modelling of undefined information. This null value is then strictly interpreted as 'not applicable'. Thus for the modelling of missing information in fuzzy databases one kind of null value is sufficient. To allow for a distinction between inapplicability in the context of different data types, the domain dom_t of each data type t is extended with an extra, domain specific null value $\perp_t$ which is used to model missing undefined data in that domain [49]. Domain specific null values allow to distinguish between undefined values of different data types in such a way that the available information about the data type of the record field is not neglected. As such, for example, inapplicability where one expects an integer value differs from inapplicability where one expects a character string value, and will explicitly be modelled by the null values $\perp_{INTEGER}$ and $\perp_{STRING}$.

To obtain an adequate logic for the handling of null values, a three-valued logic with truth values 'true' (T), 'false' (F) and undefined ($\perp$) can be enriched with possibilistic uncertainty. This is the approach taken in the development of extended possibilistic truth values (EPTVs) [20] (cf. subsection 2.4.2).

When we consider domain specific null values, three linguistic terms 'UNK', 'UNA' and 'N/A' can be defined for each domain dom_t. These are respectively modelled by:

- The term 'UNK' stands for 'unknown (but applicable)' and corresponds with the rectangular possibility distribution

$$\begin{aligned}\pi_{UNK}(x) &= 1, \text{ iff } x \in dom_t \setminus \{\perp_t\}\\ &= 0, \text{ iff } x = \perp_t\end{aligned}$$

- The term 'N/A' stands for 'not applicable' and corresponds with the possibility distribution

$$\begin{aligned}\pi_{N/A}(x) &= 0, \text{ iff } x \in dom_t \setminus \{\perp_t\}\\ &= 1, \text{ iff } x = \perp_t\end{aligned}$$

- The term 'UNA' stands for 'not available' and corresponds with the possibility distribution

$$\pi_{UNA}(x) = 1, \forall x \in dom_{A_i}$$

These possibility distributions are visualized in figure 4.2.

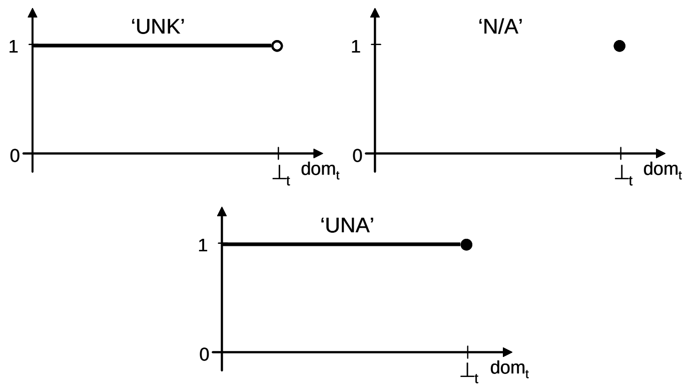

Fig. 4.2 Linguistic terms for the modelling of missing information.

The use of a special null value for the handling of undefined information still brings along with it a problem of incomplete truth functionality [16]. Indeed, if we use extended possibilistic truth values (EPTVs), we are still not able to adequately deal with the two special cases $\tilde{t}^*(p\ AND\ NOT\ p)$ and $\tilde{t}^*(p\ OR\ NOT\ p)$. In order to have correct query results in these cases too, the database system has to handle them explicitly.

4.1.4 Dealing with inconsistent information in databases

In order to obtain a consistent database, some integrity rules must be defined on the database. This could be done by means of **constraints** [41]. Furthermore, there exist more advanced techniques to detect and to solve inconsistencies in databases. Their description falls outside the scope of this book.

4.2 Database modelling

The techniques described in the previous section 4.1 form the basis of several advanced database models for the management of imperfect information. Each of these models is an extension or generalization of a traditional database model. As already mentioned at the beginning of this chapter there exist extensions and/or generalizations of the relational database model, the (E)ER-model, object relational models and object oriented models. In what follows, the modelling aspects of four representative 'fuzzy' database models are described in more detail. The first and second are based on the relational database model, the third and fourth are based on an object oriented database model. The data manipulation and querying aspects are described in chapter 5.

4.2.1 Possibilistic approaches

The possibilistic relational model, as presented in [46], is a generalization of the relational model [12]. For the generalization, possibility theory is used [62].

At the basis of the model is the generalization of the domains of the data types that are associated with the attributes of a relation. For the generalization, these domains are firstly extended with an extra domain value $\perp$ –which is used to denote that the attribute is not applicable for the tuple– and then secondly generalized to a set of fuzzy sets which all characterize possibility distributions. Thus, for each database relation with schema

$$R(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$$

with attributes $A_i : T_i$, $1 \leq i \leq n$ the domains dom_{T_i} are generalized to

$$\tilde{dom}_T = \tilde{\wp}(dom_{T_i} \cup \{\perp\})$$

where each element of the new domain $\tilde{dom}_T$ characterizes a possibility distribution π_{A_i}.

Example 4.5
As an example consider a database in which information about company cars is registered. As such, for each employee, among others, the age of his/her company car is stored. With respect to the age of Peter's company car, the following situations are possible:

- *We don't know if Peter has a company car and if he has one, we don't know the age of this car. This can be modelled by the domain value*

$$\pi_{Car\_age(Peter)}(x) = 1, \forall\, x \in dom_T \cup \{\perp\}$$

where T is the data type of the attribute Car_age.
- *It is completely possible that Peter has no company car, but it is also possible with a possibility $\lambda > 0$ that Peter has a company car that is more than five years old. This can be modelled by the domain value*

$$\pi_{Car\_age(Peter)}(x) = \begin{cases} 1 & \text{iff } x = \perp \\ \lambda & \text{iff } x > 5 \\ 0 & \text{iff } x \leq 5 \end{cases}$$

- *It is completely certain that Peter has no company car. This can be modelled by the domain value*
$$\pi_{Car\_age(Peter)}(x) = \begin{cases} 1 & \text{iff } x = \perp \\ 0 & \text{otherwise} \end{cases}$$
- *It is completely possible that Peter has a new company car, although there is also a possibility $\lambda > 0$ that he has none. This can be modelled by the domain value*
$$\pi_{Car\_age(Peter)}(x) = \begin{cases} \lambda & \text{iff } x = \perp \\ \mu_{new}(x) & \text{otherwise} \end{cases}$$
where μ_{new} is the membership function that is used to represent the fuzzy predicate 'new'.
- *It is completely certain that Peter has a company car, but there is no information available about the age of this car. This can be modelled by the domain value*
$$\pi_{Car\_age(Peter)}(x) = \begin{cases} 0 & \text{iff } x = \perp \\ 1 & \text{otherwise} \end{cases}$$
- *It is completely certain that Peter has a company car which is between two and four years old. This can be modelled by the domain value*
$$\pi_{Car\_age(Peter)}(x) = \begin{cases} 1 & \text{iff } x \in [2,4] \\ 0 & \text{otherwise} \end{cases}$$
- *It is completely certain that Peter has a new company car. This can be modelled by the domain value*
$$\pi_{Car\_age(Peter)}(x) = \begin{cases} 0 & \text{iff } x = \perp \\ \mu_{new}(x) & \text{otherwise} \end{cases}$$
- *It is completely certain that Peter has a two years old company car. This can be modelled by the domain value*
$$\pi_{Car\_age(Peter)}(x) = \begin{cases} 1 & \text{iff } x = 2 \\ 0 & \text{otherwise} \end{cases}$$

◇

All possibility distributions are assumed to be normalized. This corresponds to natural expectations because $dom_{T_i} \cup \{\perp\}$ describes all the possible alternatives. Furthermore, in the approach it is assumed that all attributes are single-valued. This conforms with the atomicity property of attributes in the relational database model.

The uncertainty on the data in the database —data modelled by a normalized possibility distribution which is not restricted to a singleton— propagates to the manipulation and querying of the database. In order to model that a tuple does not belong with complete certainty to a query result, all database (and result) tables are extended with two extra columns ‘Pos’ (possibility) and ‘Nec’ (necessity) which contain membership grades. In this way, two fuzzy sets are defined on the rows of each table. Because of the fact that column definitions of a table could be seen as a predicate, these fuzzy sets respectively denote which rows possibly satisfy this predicate and which rows necessarily satisfy this predicate. The membership grades then respectively correspond with the values of a possibility measure and the values of a necessity measure.

In the model, the operators of the relational algebra are all generalized. This is done in such a way that each operator additionally computes the associated values for Pos and Nec for each row in its result set. This guarantees the closeness property of the generalized relational algebra. Querying of ‘fuzzy’ databases is described in chapter 5.

Example 4.6
Generalizing the relation ‘Painting’ in the ‘artworks’-database of example 3.1 by means of the possibilistic relational database model results for example in the ‘fuzzy’ relation that is represented in table 4.1 (for the sake of the representation, the attribute ‘Name’ has been omitted). The attributes ‘Period’, ‘Value’ and ‘Owner’

Table 4.1 ‘Fuzzy’ relation ‘Painting’.

PID	Artist	Period	Value	Owner	Pos	Nec
P01	A04	1882	about_15M	$\{(\text{Boijmans}, 1), (\text{KMSK}, 0.6)\}$	1	1
P02	A02	around_1870	more_than_6M	$\{(\text{Louvre}, 1)\}$	1	1
P03	A01	very_old	very_expensive	$\{(\text{Louvre}, 1)\}$	1	1
P04	A03	1881	at_least_100K	$\{(\text{KMSK}, 1)\}$	1	1

have a generalized data type. The values of the attributes ‘Period’ and ‘Value’ are represented by linguistic terms which are all modelled by a possibility distributions. Remark that attributes that belong to a candidate key are not allowed to contain imprecise or uncertain values. Attributes that belong to a foreign key can have an uncertain value (e.g., attribute ‘Owner’). It is assumed by default that all rows certainly belong to the table, i.e., for each row $\text{Pos} = 1$ *and* $\text{Nec} = 1$. $\diamond$

4.2.2 Similarity relation based approaches

An alternative for the possibilistic model is the **similarity based** relational model as originally presented in [10]. In this approach the domains dom_T of the data types T that are associated with the attributes of a relation are extended with a *similarity relation* S_T (cf. subsections 3.2.1.3 and 4.1.1) and generalized to

$$\tilde{dom}_T = \{V|V \in \wp(dom_T) \wedge V \text{ is a finite discrete set}\} \cup \{\perp_{null}\}$$

where $\wp(dom_T)$ denotes the power set of dom_T and the extra domain value $\perp_{null}$ is used for the modelling of both unknown and undefined information. The set dom_T is called the basic set of the domain.

Thus for each database relation with schema

$$R(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$$

with attributes $A_i : T_i$, $1 \leq i \leq n$ the domains dom_{T_i} of the attributes are generalized to a couple

$$(\tilde{dom}_{T_i}, S_{T_i}).$$

Example 4.7
As an example, one can consider a generalized domain that is associated with an extra attribute $Dominant\_Colour$ that is added to the schema of the relation 'Painting' of the 'artworks'-database of example 3.1. This domain is for example defined as

$$(\tilde{dom}_T, S_T)$$

where the basic set dom_T is scalar and defined by

$$dom_T = \{white, green, blue, red\}$$

and the associated similarity relation S_T is given by the similarity matrix that is presented in table 4.2. Remark that $white$ is only considered to be similar with itself.

Table 4.2 Similarity relation S_T.

S_T	*white*	*green*	*blue*	*red*
white	1	0	0	0
green	0	1	0.5	0.3
blue	0	0.5	1	0.3
red	0	0.3	0.3	1

◇

Due to the generalization of the domains, each tuple t_i, $1 \leq i \leq m$ of a relation

$$R(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n]$$

has the following form

$$(A_1 : W_{i,1}, A_2 : W_{i,2}, \ldots, A_n : W_{i,n})$$

where $W_{i,j} \subseteq dom_{T_j}$, $1 \leq j \leq n$. The possible interpretations

$$\alpha = [a_1, a_2, \ldots, a_n]$$

of a tuple are obtained by taking an element a_j in each of the sets $W_{i,j}$. Also in this approach, each tuple is interpreted as being a proposition that should satisfy the predicate imposed by the relation schema. Thus, in order to be a valid interpretation, the interpretation must satisfy the predicate imposed by the relation schema.

Furthermore, the database model prescribes that a *similarity threshold value* is associated with each attribute $A : T$ of a relation. This threshold value is by definition obtained as:

$$THRES(A) = \min_{V \in t[A]} (\min_{x,y \in V} (S_T(x,y)))$$

where $t[A]$ denotes the set of all values (sets) V $(\in \tilde{dom}_T)$ of the attribute with name A that occur in a tuple t of the relation. For regular databases the cardinality of V is always equal to 1 and $S_T(x,y) = 1$, such that $THRES(A) = 1$, for all attributes A of the relation. These similarity threshold values play a role in the database querying.

For querying purposes, the operators of the relational algebra are extended in such a way that the user can specify a minimal similarity threshold value for each of the attributes involved in the query. The result of the query is then built up in such a way that no extra tuples can be added without exceeding at least one of the minimal threshold values. If the user did not specify threshold values then these are implicitly considered to be equal to 1. Threshold values could also be represented by linguistic terms in which case each linguistic term is associated with an exact threshold value. Querying in similarity relation based approaches is further described in section 5.4.

Example 4.8
Using the similarity based relational model, the relation 'Owner' of the 'artworks'-database of example 3.1 can be generalized to a 'fuzzy' relation as presented in table 4.3. For this relation, the following algebraic expression can be considered

Table 4.3 'Fuzzy' relation 'Owner'.

Name	*Place*	*Country*
{Boijmans}	{Rotterdam,Amsterdam}	{The Nederlands,Holland}
{Louvre}	{Paris}	{France}
{KMSK}	{Antwerp,Mechlin,Bornem}	{North Belgium,Flanders}

$$\begin{aligned} &project(Owner, \{Place, Country\}) \\ &\qquad WITH\ THRES(Place) \geq 0.75 \\ &\qquad AND\ THRES(Country) \geq 0.80 \end{aligned}$$

The result of this operation consists of a new relation which is obtained by removing the attribute 'Name'. In order to obtain the resulting tuples, the tuples of the remaining intermediate relations are combined, if possible. The idea behind this is *that in a regular database it is not allowed to have identical tuples in the same relation and identity is now 'weakened' to similarity. Two tuples can be combined if application of the union operator for sets does not result in a crossing of one of the given threshold values.* $\diamond$

A similar, similarity based object oriented database model has been presented in [29].

4.2.3 The fuzzy object oriented FOOD database model

The FOOD-model [5, 36, 6, 7] is an extension of the 'on graphs based' **object model** of Lucarella [37], where as well the *database schema* as its *instances* are represented by directed labelled graphs and where the database manipulations are defined in terms of graph transformations.

The conceptual schema Σ of Lucarella's model is defined by a quintuple

$$(C,T,A,H,P)$$

where:

- C is a finite set of *(object) classes*;
- T is a finite set of primitive *types*. Each type $t \in T$ has an associated set $V(t)$ of allowed values;
- A is a finite set of *attributes*. Each attribute has an associated domain. The domain of a primitive attribute is a primitive type $t \in T$, the domain of a complex attribute is a *class* $c \in C$;
- $H \subseteq C \times C$ is the *inheritance relation*. This is a partial ordering relation, where $(c_i, c_j) \in H$ means that c_i is a subclass of c_j;
- $P \subseteq C \times A \times (C \cup T)$ is the *property relation* which denotes the attributes that are associated with a class. If $(c_i, a, c_j) \in P$, this means that c_i has an attribute a with domain c_j.

The conceptual schema Σ is represented by means of a directed labelled graph

$$G(\Sigma) = (N,E)$$

where:

- $N = C \cup T$ is the set of all nodes of the graph. Each class $c \in C$ is represented by a rectangular node with label c, each (primitive) type $t \in T$ is represented by an oval node with label t;
- $E = H \cup P$ is the set of all labelled edges of the graph. Each element $(c_i, c_j) \in H$ is represented by an arrow that is directed from c_i to c_j and has a label 'is a', whereas each element $(c_i, a, c_j) \in P$ is represented by an arrow that is directed from c_i to c_j and has a label a.

An object oriented database S is finally defined by a quadruple

$$(\Sigma, O, I, L)$$

where:

- Σ is the conceptual schema that has been defined above;
- O is the set of all objects that are present in the database; $V(T)$ is the (sub)set of all primitive values: $V(T) = \bigcup_{t \in T} V(t)$;
- $I \subseteq (O \times C) \cup (V(T) \times T)$ is the instance relation; each object $o \in O$ is an instance of a class $c \in C$ and each value $v \in V(T)$ is an instance of a primitive type $t \in T$;
- $L \subseteq O \times A \times (O \cup V(T))$ is the reference relation; $(o_i, a, o_j) \in L$ means that o_j is the value for a in the *object* o_i.

The instance relation I allows it to construct an instance graph for a schema graph $G(\Sigma)$. At each time, the nodes of an instance graph correspond to the objects and values that are present in the database.

For the FOOD-model the following extensions have been established:

1. Definition of *fuzzy attribute values*. Fuzzy attribute values are modelled by means of possibility distributions that are defined on the domain of the attribute. For this reason, a so-called set of *fuzzy types* T_v is added. This is done by extending the set of primitive types to an extended set $T_e = T \cup T_v$. For this set it holds that: $V(T_e) = V(T) \cup V(T_v)$.
2. Definition of *fuzzy classes*. A so-called set of fuzzy classes C_f is introduced. This is done by extending the set of classes C to an extended set $C_e = C \cup C_f$.
3. Definition of *uncertain property and reference relations*. Uncertainty about the association of a value to an attribute of an object is formalized by means of the uncertain property relation

   $$P_u \subseteq C_e \times A \times (C_e \cup T_e)$$

   and the uncertain reference relation

   $$L_u : O_e \times A \times (\wp(O_e) \cup \wp(V(T_e))) \to [0,1]$$

   where O_e is the extended counterpart of the set O.
4. Definition of *'strengthened' (imprecise) property and reference relations*. Independently of dealing with uncertainty it can also happen that we have to deal with imprecision when associating a value to an attribute of an object. This is formally dealt with by means of the 'strengthened' property relation

   $$P_s \subseteq C_e \times A \times (C_e \cup T_e)$$

   and the 'strengthened' reference relation

   $$L_s \subseteq O_e \times A \times (\wp(O_e \times T(Strength)) \cup \wp(V(T_e) \times T(Strength)))$$

where $T(Strength) = \{none, very\ low, low, high, very\ high, full\}$ is the set of the allowed linguistic terms for the linguistic variable $Strength$. Imprecision can also occur within the elements of an uncertain property relation or an uncertain reference relation. Therefore the uncertain ‘strengthened’ property relation

$$P_{su} \subseteq C_e \times A \times (C_e \cup T_e)$$

and the uncertain ‘strengthened’ reference relation

$$L_{su} : O_e \times A \times (\wp(O_e \times T(Strength)) \cup \wp(V(T_e) \times T(Strength))) \rightarrow [0,1]$$

are introduced.

5. Definition of the *‘strengthened’ (imprecise) instance relation*. This concept is introduced to make it possible that an object is only partly (to a given extent) an instance of a class and is defined by

$$I_f \subseteq O_e \times C_f \times T(Strength).$$

6. Definition of *fuzzy class hierarchies*. An extended inheritance relation

$$H_f \subseteq C_f \times C_f \times Modifiers$$

allows it to express, by means of the set $Modifiers$ of modifiers, to which extent an object of a subclass belongs to a superclass.

These extensions are used for the formal definition of the extended conceptual schema Σ_e and the so-called fuzzy object oriented multimedia system M which can be graphically represented by means of an adapted representation [6, 7].

4.2.4 The constraint based object oriented database model

The fourth ‘fuzzy’ database model that is described in more detail in this chapter is based on *generalized constraints*, as introduced by L.A. Zadeh (cf. subsection 2.5) [23]. Generalized constraints can be used to specify the semantics and integrity of the databases and to state the selection criteria in queries. As underlying logical framework, a logic based on *extended possibilistic truth values* (EPTVs) is used (cf. subsection 2.4.2.2). Furthermore, the model is, as far as its regular components are considered, in conformity with the *ODMG data model* [11].

The basis of the database model is formed by a so-called type system —that states the supported data type definitions— and a so-called constraint system —that states the allowed constraint definitions. Starting from data types and constraints, object schemas and database schemas are specified. The database model is completed with data definition and data manipulation operators.

4.2.4.1 Data types and type system

Like in the ODMG data model [11], the common characteristics of objects are specified by class properties which on their turn are built up of **data types**. Each data type t is a generalization of an ODMG data type and is among others characterized by a domain dom_t and a set of operators O_t which all operate on the domain values of the data type. Each domain contains a domain specific value $\perp_t$ which is used to represent 'undefined' information (cf. subsection 4.1.3.2). The data types supported by the database model are defined by a *type system*. To be conform with the ODMG data model, the type system supports definitions for *literal types* ($\in T_{literal}$) and *object types* ($\in T_{object}$). Additionally, there are also definitions for *reference types* ($\in T_{reference}$). Reference types allow it to refer to instances of object types and are used to model the binary relationships that can exist between object types in a database schema.

The **reference types** are subdivided in:

- *Single-valued reference types*, which are denoted by $Ref(t)$ where t is an object type.
- *Multiple-valued reference types*, which are denoted by $Ref_{Set}(t)$, $Ref_{Bag}(t)$ or $Ref_{List}(t)$ where t is an object type.

The **literal types** are subdivided in:

- *Basic types*, like e.g. $Integer$, $Real$, $Boolean$, and $String$.
- *Collection types*, like e.g. $Set(t)$, $Bag(t)$, $List(t)$, and $Array(t)$ where t is a literal type.
- *Structured types*. In general a structured type is specified as

  $$Struct\ id(id_1\ isr_1\ t_1; id_2\ isr_2\ t_2; \ldots; id_n\ isr_n\ t_n)$$

  where id is the name of the type and

  $$(id_1\ isr_1\ t_1, id_2\ isr_2\ t_2, \ldots, id_n\ isr_n\ t_n)$$

  are the components of the type. Each component $id_i\ isr_i\ t_i$, $1 \leq i \leq n$ represents a (generic) generalized constraint that acts on a variable id_i with associated data type $t_i \in T_{literal} \cup T_{reference}$. The variable copula isr_i can take the following values $isr_i \in \{ise, is, isv\}$. The semantics of these constraints can be described in a simplified way as:

  – If $isr_i = ise$, then the allowed values for id_i are restricted to the values of the domain dom_{t_i} of the associated data type t_i of id_i.
  – If $isr_i = is$, then id_i is interpreted as a (disjunctive) possibilistic variable. The allowed values for id_i are in this case restricted to the fuzzy sets that are defined over the domain dom_{t_i}. The membership grades of these fuzzy sets are interpreted as degrees of uncertainty. With other words, such a fuzzy set represents a possibility distribution.

– If $isr_i = isv$, then id_i is interpreted as a (conjunctive) veristic variable. The allowed values for id_i are in this case restricted to the fuzzy sets that are defined over the domain dom_{t_i} and of which the membership grades are interpreted as degrees of truth.

With a view on the specification of the **object types** the concept operator signature is introduced. In general, an operator signature is specified as:

- $Signat(( ) \rightarrow t')$
- $Signat((id'_1\ isr_1\ t'_1; id'_2\ isr_2\ t'_2; \ldots ; id'_p\ isr_p\ t'_p) \rightarrow t')$

Hereby, t' is the data type of the result values. It holds that $t' \in T_{literal} \cup T_{reference} \cup \{Void\}$ with $Void$ being a data type that is used in situations where no further, more specific, type specification can be given [11] (which might, e.g., be the case if the operator produces no results). Furthermore, $id'_i\ isr_i\ t'_i$, $1 \leq i \leq p$ are the parameters of the operator. Each parameter is represented by a (generic) generalized constraint that acts on a variable id'_i that has to take values of the domain of the associated data type t'_i. Again, the variable copula isr_i is allowed to one of the following values $isr_i \in \{ise, is, isv\}$ where the different options are interpreted as described above.

In general, an object type is specified as follows:

- $Class\ id(id_1\ isr_1\ s_1; id_2\ isr_2\ s_2; \ldots ; id_n\ isr_n\ s_n)$
- $Class\ id : \widehat{id}_1, \widehat{id}_2, \ldots , \widehat{id}_m( )$
- $Class\ id : \widehat{id}_1, \widehat{id}_2, \ldots , \widehat{id}_m(id_1\ isr_1\ s_1; id_2\ isr_2\ s_2; \ldots ; id_n\ isr_n\ s_n)$

Hereby, id denotes the name of the object type, the identifiers $\widehat{id}_i$, $1 \leq i \leq m$ indicate the parent types of the object type (if these exist) —hereby the ODMG type-subtype inheritance mechanism is considered [11]— and

$$(id_1\ isr_1\ s_1; id_2\ isr_2\ s_2; \ldots ; id_n\ isr_n\ s_n)$$

are the *characteristics* of the object type. A characteristic $id_i\ isr_i\ s_i$, $1 \leq i \leq n$ is:

- an *attribute* if s_i is a literal type or an object type;
- a *(binary) relationship* if s_i is a reference type;
- a *method* if s_i is an operator signature.

Each characteristic is represented by a (generic) generalized constraint that acts on a variable id_i with associated specification s_i. The allowed copula are also in this case $isr_i \in \{ise, is, isv\}$ with the same semantics as explained above. For methods, the generalized constraints puts a restriction on the allowed result values of the operator. Beside the characteristics that are explicitly represented in the specification of the object type, an object type also inherits the characteristics of its parent types $\widehat{id}_i$, $1 \leq i \leq m$ (if these exist and are specified in the object type specification).

The type system TS, which defines all data types that are supported by the database model, is defined as a quadruple:

$$TS \equiv [ID, T, \leftrightarrow, \prec]$$

where

- ID is the set of all valid identifiers,
- T is the set of all supported type specifications, i.e.

$$T \equiv \{Void\} \cup T_{reference} \cup T_{literal} \cup T_{object}$$

- $\leftrightarrow$: $T_{object} \times T_{object} \rightarrow \{True, False\}$ is a partial relation that defines the binary (association) relationships between object types.
- $\prec$: $T_{object} \times T_{object} \rightarrow \{True, False\}$ is a partial ordering relation that defines the on inheritance based type-subtype relationships between object types.

Example 4.9
The type system TS allows it to specify the following (simplified) types for the modelling of employees. With the structured types

- $Struct\ TAddress(Street\ ise\ String; City\ ise\ String)$
- $Struct\ TCompany(Name\ ise\ String; Location\ ise\ String)$
- $Struct\ TWorks(Company\ ise\ TCompany; Percentage\ is\ Real)$

and the enumeration literal type

$$Enum\ TLanguage(French, English, Spanish, Italian)$$

the object types $TPerson$ and $TEmployee$ can be defined as follows:

$$\begin{aligned}
Class\ TPerson(&Name\ ise\ String;\\
&Year\_of\_birth\ is\ Integer;\\
&Address\ ise\ TAddress;\\
&Languages\ isv\ TLanguage;\\
&Children\ ise\ Set_{Ref}(TPerson);\\
&Add\_child\ ise\ Signat((New\_child\ ise\ TPerson) \rightarrow Void))
\end{aligned}$$

$$\begin{aligned}
Class\ TEmployee : TPerson(&EmployeeID\ ise\ String;\\
&Works\_for\ ise\ Bag(TWorks))
\end{aligned}$$

$\diamond$

The instances of the reference types and literal types are respectively called **reference instances** and **literals**. These are defined as a couple $[t, v]$, where t is a reference type or a literal type and $v \in dom_t$.

Depending on its lifetime an object is either transient or persistent. Transient objects are not stored in a database. They only temporarily exist, as long as the application that created them runs. Persistent objects are stored in a database and remain in the database until they are explicitly removed by a user or application program. A **transient object** is defined by a triplet

$$o \equiv [t, v, \tilde{t}^*(\text{‘}o \text{ is an instance of } t\text{’})]$$

where

- $t \in T_{object}$ is the type of the object,
- $v \in dom_t$ is the state of the object and
- $\tilde{t}^*$(‘o is an instance of t’) is the EPTV that expresses the truth value of the proposition ‘o is an instance of thet objecttype t’.

A **persistent object** is defined as a quintuple

$$o \equiv [oid, N, t, v, \tilde{t}^*(\text{‘}o \text{ is an instance of } t\text{’})]$$

where

- $t \in T_{object}$ is the type of the object,
- $v \in dom_t$ is the state of the object,
- oid is a unique object identifier,
- N is a (finite) set of object names which act as access points to the object in the database and
- $\tilde{t}^*$(‘o is an instance of t’) is the EPTV that expresses the truth value of the proposition ‘o is an instance of the object type t’.

The unicity of the object identifier must be enforced over the complete database. The object identifier oid is used to refer to (the state of) the object. The set of object names N can be empty.

The set of all instances of an object type $t \in T_{object}$ is denoted as $V_t^{instance}$. If t is a subtype of another object type $\widehat{t}$, then it holds that $V_t^{instance} \subseteq V_{\widehat{t}}^{instance}$. The extent of an object type t is denoted as V_t^{extent} and defined as the set of all persistent objects of t that occur in a given database. If t is a subtype of another object type $\widehat{t}$, then it holds that $V_t^{extent} \subseteq V_{\widehat{t}}^{extent}$.

Example 4.10
The instances of the object type $TPerson$ of example 4.9 are either $TPerson$ objects, or $TEmployee$ objects (because $TEmployee$ is a subtype of $TPerson$). Examples of persistent $TPerson$ objects are:

$$\begin{aligned}
&[oid_1, \{\}, TPerson,\\
&\quad (Name\ ise\ \text{‘}Ann\text{’};\\
&\quad Year\_of\_birth\ is\ Around\_1993;\\
&\quad Address\ ise\ (Street\ ise\ \text{‘Church street, 12’}; City\ ise\ \text{‘Paris’});\\
&\quad Languages\ isv\ \{(French, 1), (English, 0.4)\};\\
&\quad Children\ ise\ Set()), \{(T, 1)\}]
\end{aligned}$$

and

$$
\begin{aligned}
&[oid_2, \{\}, TPerson,\\
&\quad (Name\ ise\ \text{‘}Tom\text{’};\\
&\quad Year\_of\_Birth\ is\ Around\_1990;\\
&\quad Address\ ise\ (Street\ ise\ \text{‘}Church\ street,\ 12\text{’}; City\ ise\ \text{‘}Paris\text{’});\\
&\quad Languages\ isv\ \{(French, 1), (English, 0.5), (Spanish, 0.7)\};\\
&\quad Children\ ise\ Set()), \{(T, 1)\}]
\end{aligned}
$$

An example of a persistent TEmployee object is:

$$
\begin{aligned}
&[oid_3, \{\}, TEmployee,\\
&\quad (Name\ ise\ \text{‘}Johan\text{’};\\
&\quad Year\_of\_birth\ is\ \{(1965, 1)\};\\
&\quad Address\ ise\ (Street\ ise\ \text{‘}Church\ street,\ 12\text{’}; City\ ise\ \text{‘}Paris\text{’});\\
&\quad Languages\ isv\ \{(French, 1), (Spanish, 0.8), (English, 1)\};\\
&\quad Children\ ise\ Set(oid_1, oid_2);\\
&\quad EmployeeID\ ise\ \text{‘}ID25\text{’};\\
&\quad Works\_for\ ise\ Bag((Company\ ise\ (Name\ ise\ \text{‘}XYZ\text{’};\\
&\qquad\qquad Location\ ise\ \text{‘}Paris\text{’});\\
&\qquad Percentage\ is\ \{(100, 1)\}))), \{(T, 1)\}]
\end{aligned}
$$

◇

4.2.4.2 Constraints and constraint system

Constraints can formally be seen as relations that must be satisfied. With respect to databases, constraints are an important and adequate means to define the data(base) semantics and to guarantee the data(base) integrity [34]. As such, constraints can be used to specify the semantics of an object type. An object type for the modelling of persons can then be extended with constraints that define the full semantics of the allowed domain values of attributes like ‘Year_of_birth’, ‘Height’, and ‘Weight’ of persons. Other constraints can specify the valid transitions, from the old to the new value, of attributes like ‘Salary’ (e.g. a salary can never decrease) and ‘Marital status’ (e.g. one can never be divorced without first being married, so a transition from ‘unmarried’ to ‘divorced’ is not allowed). Other constraints can enforce still other integrity constraints. A persistent instance of an object type then belongs to the database to the extent that all of the applicable constraints of the object type are satisfied for the instance.

Remark that in some object oriented database models there is no explicit support for constraints. In such models, the underlying assumption in that constraints should be dealt with in the implementation of the class methods and that an object attribute is only accessible via adequate methods that return (read), insert (write) or modify (overwrite) the actual attribute value. In the constraint based object oriented database model this assumption is not made and support for explicit constraint specifications is provided.

Constraints can also be used to specify selection conditions in queries. In such a case, each constraint states a condition. In order to belong to the result set of a query an instance must satisfy the constraints imposed by the query and belongs to the result to the extent that it satisfies these constraints. As such two constraints could be used to search for all persons that are around twenty years old and live in the environment of Paris: a constraint to find all persons that around twenty years old and a constraint to find all persons that live in the environment of Paris.

Each **constraint** is characterized by a specification and a logical function that acts on an object and results in an EPTV. The constraints supported by the database model are fully determined by a *constraint system*. To make a distinction between the (generic) generalized constraints that are used in the type specifications, the constraints that are derived from the constraint system are called specific constraints (or constraints for short).

In the constraint system four categories of constraints are being distinguished. Hereby, a first distinction is made on the basis of whether the constraint is defined over the instances of *one single object type*, or is defined over the instances of multiple object types (single-type dependent vs. multi-type dependent). A second distinction is made on the basis of whether the *complete extent* of an object type is involved in the evaluation of the constraint, or not. The four resulting categories are respectively denoted as C_i^s, C_e^s, C_i^m and C_e^m and are described as follows:

- The set C_i^s consists of **single-type dependent constraints that are not defined over the complete extent of an object type**. Examples of constraints of this category are:
  - *'Not null' constraints* which impose that null values are not allowed for the specified attribute or relationship.
  - *Value constraints* which impose a restriction on the allowed domain values of the specified attribute or relationship. Furthermore, only one object type is involved in the specification of the constraint.
  - *Transition constraints* which impose a restriction on the allowed transitions (from the old value to the new value) of the specified attribute or relationship. Again, only one object type is involved in the specification of the constraint.
- The set C_e^s consists of **single-type dependent constraints that are defined over the complete extent of an object type**. Examples of constraints of this category are:
  - *Key constraints* which put an unicity constraint and an irreducibility constraint on a specified subset of attributes and relationships of the specified object type. Unicity means that no two objects of the extent of the object type are allowed to have the same values for each of the attributes and relationships in the specified set. Irreducibility means that no attribute or relationship is

superfluous and can be removed from the specified set without violating the unicity constraint.
  - *Aggregation constraints* which impose a condition that contains at least one aggregation operator that acts on the specified object type. Only one object type is involved in the specification of the constraint.

- The set C_i^m consists of **multi-type dependent constraints that are not defined over the complete extent of an object type**. Examples of constraints of this category are:
  - *Value constraints* which impose a restriction on the allowed domain values of the specified attribute or relationship. More than one object type is involved in the specification of the constraint.
  - *Transition constraints* which impose a restriction on the allowed transitions (from the old value to the new value) of the specified attribute or relationship. Again, more than one object type is involved in the specification of the constraint.
- The set C_e^m consists of **multi-type dependent constraints that are defined over the complete extent of an object type**. Examples of such constraints are:
  - *Unicity constraints* which put an unicity constraint on the object identifiers and object names that are associated with objects in the extents of the specified object types. No two object in the union of these extents are allowed to have the same object identifier or object name.
  - *Referential constraints* which guarantee referential integrity for the specified relationships. Referential integrity imposes that that the objects that are referred to must always exist in the database.
  - *Aggregation constraints* which impose a condition that contains at least one aggregation operator that acts on at least one of the specified object types. More than one object type is involved in the specification of the constraint.

The constraint system *CS*, which defines all valid (explicit) constraints that are supported by the database model, is defined as a triple:

$$CS \equiv [ID, E, C]$$

where

- *ID* is the set of all valid identifiers,
- *C* is the set of all supported constraint specifications, i.e.

$$C \equiv C_i^s \cup C_e^s \cup C_i^m \cup C_e^m$$

- *E* is the set of all valid characterizing logical functions for constraints. Each logical function maps an object of each of the involved object types onto an EPTV, which denotes to which extent the objects satisfy the constraint.

Example 4.11
With respect to the object types $TPerson$ and $TEmployee$ that have been introduced in example 4.9, the following constraints can be considered:

- $c_1 = c^{not\_null}_{\{TEmployee.EmployeeID\}}[\,]$
- $c_2 = c^{value}_{\{TEmployee.Year\_of\_birth\}}[\,around\_1930 \leq TPerson.Year\_of\_birth \leq 1990\,]$
- $c_3 = c^{value}_{\{TEmployee.Works\_for.Percentage\}}[\,0 \leq TEmployee.Works\_for.Percentage \leq 100\,]$
- $c_4 = c^{key}_{\{TPerson\}}[\,TPerson.Name\,]$
- $c_5 = c^{oid}_{\{TPerson,\{TPerson,TEmployee\}\}}[\,]$
- $c_6 = c^{name}_{\{TPerson,\{TPerson,TEmployee\}\}}[\,]$
- $c_7 = c^{oid}_{\{TEmployee,\{TPerson,TEmployee\}\}}[\,]$
- $c_8 = c^{name}_{\{TEmployee,\{TPerson,TEmployee\}\}}[\,]$
- $c_9 = c^{reference}_{\{TPerson.Children\}}[\,]$

Hereby, c_1 is a 'Not null' constraint, c_2 and c_3 are value constraints, c_4 is a key constraint, c_5, c_6, c_7 and c_8 are unicity constraints and c_9 is a referential constraint. $\diamond$

4.2.4.3 Object schemas and database schemas

The semantics of an object is described by its **object schema**. This schema completely defines the object and contains the definitions of all specific constraints that are defined for the object type of the object. Each object schema os is defined as a quadruple

$$os = [id, t, M, C_t]$$

where

- $id \in ID$ represents the name of the object schema.
- $t \in T_{object}$ is the type of the object schema.
- M describes the 'meaning' of the object schema. M is provided to add comments and is usually a description in natural language.
- $C_t \in \tilde{\wp}(C^s_i)$ is a normalized fuzzy set of explicit constraints that all have to act on the objects of type t. The membership grades of C_t are interpreted as weights and denote the relative importance of the constraints within the object schema.

The set of all existing object schemas is denoted as OS and is defined to be the set of all quadruples that satisfy the above definition.

An instance o of the object type t is by definition also an instance of the object schema $os = [id, t, M, C_t]$, if and only if it satisfies (with a EPTV that differs from $\{(F,1)\}$) all constraints of C_t and all constraints of the fuzzy sets $\widehat{C_{\hat{t}}}$ of the object schemas

$$[\widehat{id}, \widehat{t}, \widehat{M}, \widehat{C_{\hat{t}}}]$$

that are defined for the super types $\widehat{t}$ of t. In this way, inheritance has an impact on the constraints that has to be satisfied. The set of all instances of an object schema os is denoted as $V_{os}^{instance}$, while the set of all persistent instances of os is denoted as V_{os}^{extent}. Clearly, it holds that $V_{os}^{instance} \subseteq V_t^{instance}$ and $V_{os}^{extent} \subseteq V_t^{extent}$.

Example 4.12
With respect to the object types $TPerson$ and $TEmployee$ that have been introduced in example 4.9 and the constraints $c_1, c_2, \ldots, c_9$ from example 4.11 the following object schema's can be considered:

$$OSPerson = [OSPerson, TPerson, \text{‘schema to represent person objects’}, \{(c_2, 1)\}]$$

and

$$OSEmployee = [OSEmployee, TEmployee, \text{‘schema to represent employee objects’}, \{(c_1, 1), (c_3, 0.7)\}]$$

$\diamond$

A **database schema** describes the semantics of the objects stored in a database. Each database schema ds is defined as a quadruple

$$ds = [id, D, M, C_D]$$

where

- $id \in ID$ is the name of the database schema.
- $D = \{os_1, os_2, \ldots, os_n\} \subset OS \setminus \{\bot_{OS}\}$ is a finite set of object schemas. Each object schema in D has a different object type. If an object schema $os \in D$ is defined for an object type t and t' is a super type of t or t' is an object type that contains a binary relationship that refers to t, then D must contain an object schema $os' \in D$ that is defined for t'.
- M describes the ‘meaning’ of the database schema.
- $C_D \in \tilde{\wp}(C_e^s \cup C_i^m \cup C_e^m)$ is a normalized fuzzy set of constraints that all put extra restrictions on the instances of the object schemas of D. The membership grades of C_D are interpreted as weights and denote the relative importance of the constraints within the database schema. For each object schema $os \in D$ there exist unicity constraints in C_D which guarantee the unicity of the object identifiers and object names of the instances of os. Furthermore, each constraint $c \in C_e^s \cup C_e^m$ must be defined for the extent of the object type t of an object schema $os \in D$.

The set of all existing database schemas is denoted as DS and is defined as the set of all quadruples that satisfy the above definition.

Each persistent instance o of an object schema $os \in D$ of a database schema ds must satisfy (with an EPTV that differs from $\{(F, 1)\}$) all constraints in C_D. An instance of a database schema ds is called a database and is defined as the set of the extents of all object schemas of ds. In this way, each database is a set of sets of objects.

Example 4.13
With the object schemas OSPerson and OSEmployee of example 4.12 and the constraints $c_1, c_2, \ldots, c_9$ from example 4.11 the following database schema can be constructed:

$$\begin{aligned} DSEmployee = [&DSEmployee, \{OSPerson, OSEmployee\}, \\ &\text{'schema for an employee database'}, \\ &\{(c_4,1),(c_5,1),(c_6,1),(c_7,1),(c_8,1),(c_9,1)\}] \end{aligned}$$

By only considering the object identifiers of the persistent objects of example 4.10, the corresponding database can be represented as:

$$\{\{oid_1, oid_2, oid_3\}, \{oid_3\}\}$$

◇

4.2.4.4 Database model

Finally, the database model is obtained by considering specific operators for data definition (DDL) and data manipulation.

For data definition purposes, operators are considered to create and to remove databases and database schemas (*Create_DB*, *Drop_DB*), to add and to remove object schemas to database schemas (*Create_OS*, *Drop_OS*), to add and to remove characteristics to the object type of an object schema (*Add_Char*, *Drop_Char*), to add and to remove weighted constraints to an object schema (*Add_OSC*, *Drop_OSC*) and to add and to remove weighted constraints to a database schema (*Add_DBC*, *Drop_DBC*).

The operators for data manipulation provide facilities to add, remove, modify and query database objects. These operators all act on sets of instances that are associated with an object schema and result in a new object schema with a new associated set of instances. In this way, each data manipulation operator can act on the result of each data manipulation operator which allows to construct algebraic expressions and guarantees the closeness property of the set of data manipulation operators. The supported operators are the set operators union ($\cup$), intersection ($\cap$), difference ($\setminus$) and Cartesian product ($\otimes$), the database operators projection (Π), extension (Θ), selection (σ) and threshold (τ) and the operators for making an object persistent or transient (*Make_transient*, *Make_persistent*). The extension operator allows it to add derived attributes to the object type of an object schema. The threshold operator is used to restrict the set of instances of an object schema based on the given threshold values for EPTVs. The semantics of all other operators is analogous to that of their relational counterparts (cf. section 3.4.1.2).

The database model DM is then finally defined by:

$$DM = [TS, CS, OS, DS, O_{DDL}^{model}, O_{DML}^{model}]$$

where

- TS is the type system,
- CS is the constraint system,
- OS is the set of all object schemas,
- DS is the set of all database schemas,
- O_{DDL}^{model} is the set of all data definition operators and
- O_{DML}^{model} is the set of all data manipulation operators.

More details about this database model are described in [23].

4.2.5 Other approaches

References

1. I. Arrazola, A. Plainfossé, H. Prade, and C. Testemale, "Extrapolation of fuzzy values from incomplete data bases", *Information Systems* **14** 6 (1989) 487–492.
2. K. Atanassov, "Intuitionistic fuzzy sets", *Fuzzy Sets and Systems* **20** (1986) 87–96.
3. D. Barbara, H. Garcia-Molina, and D. Porter, "A Probabilistic Relational Data Model", in: *Proceedings of the EDBT'90 Conference* (1990) 60–74.
4. J. Biskup, "A Formal Approach to Null Values in Database Relations", in: H. Gallaire, J. Minker en J. Nicolas (eds.), *Advances in Data Base Theory*, Plenum Press, New York, USA (1981) 299–341.
5. G. Bordogna, D. Lucarella, and G. Pasi, "A Fuzzy Object Oriented Data Model", in: *Proceedings of the Third IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'94*, Orlando, USA (1994) 313–318.
6. G. Bordogna, D. Lucarella, and G. Pasi, "Extending a Graph-Based Data Model to Manage Fuzzy and Uncertain Information", in: R. De Caluwe (ed.), *Fuzzy and Uncertain Object-Oriented Databases: Concepts and Models*, World Scientific, Signapore (1997) 97–122.
7. G. Bordogna, G. Pasi, and D. Lucarella, "A Fuzzy Object-Oriented Data Model for Managing Vague and Uncertain Information", *International Journal of Intelligent Systems* **14** 7 (1999) 623–651.
8. G. Bordogna, and G. Pasi (eds.), *Recent Issues on Fuzzy Databases* (Physica-Verlag, Heidelberg, Germany, 2000).
9. P. Bosc, and J. Kacprzyk (eds.), *Fuzziness in Database Management Systems* (Physica-Verlag, Heidelberg, Germany, 1995).
10. B.P. Buckles, and F.E. Petry, "A Fuzzy Representation of Data for Relational Databases", *Fuzzy Sets and Systems* **7** (1982) 213–226.
11. R.G.G. Cattell, and D. Barry (eds.), *The Object Data Standard: ODMG 3.0* (Morgan Kaufmann Publishers, San Francisco, USA, 2000).
12. E.F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Communications of the ACM* **13** 6 (1970). Republished in *Communications of the ACM* **26** 1 (1983).
13. E.F. Codd, "RM/T: Extending the Relational Model to capture more meaning", *ACM Transactions on Database Systems* **4** 4 (1979).
14. E.F. Codd, "Missing Information (Applicable and Inapplicable) in Relational Databases", *ACM SIGMOD Record* **15** 4 (1986) 53–78.
15. C.J. Date, "Null Values in Database Management", in: *Relational Database: Selected Writings*, Addisson-Wesley Publishing Company, Reading, Massachusetts, USA (1986), 313–334.
16. C.J. Date, "NOT is Not 'Not'! (notes on three-valued logic and related matters)", in: *Relational Database Writings 1985–1989*, Addisson-Wesley Publishing Company, Reading, Massachusetts, USA (1990) 217–248.

17. C.J. Date, “EXISTS is Not ‘Exists’! (some logical flaws in SQL)”, in: *Relational Database Writings 1985–1989*, Addisson-Wesley Publishing Company, Reading, Massachusetts, USA (1990) 339–356.
18. C.J. Date, “Faults and Defaults”, in: C.J. Date, H. Darwen, and D. McGoveran (eds.), *Relational Database Writings 1994–1997*, Addisson-Wesley Publishing Company, Reading, Massachusetts, USA (1998).
19. R. de Caluwe (ed.), *Fuzzy and Uncertain Object-oriented Databases: Concepts and Models* (World Scientific, Signapore, 1997).
20. G. De Tré, “Extended Possibilistic Truth Values”, *International Journal of Intelligent Systems* **17** (2002) 427–446.
21. G. De Tré, and R. De Caluwe, “Level-2 fuzzy sets and their usefulness in object-oriented database modelling”, *Fuzzy Sets and Systems* **140** (2003) 29–49.
22. G. De Tré, R. De Caluwe, J. Kacprzyk, and S. Zadrozny, “On flexible querying via extensions to fuzzy sets”, in: *Proceedings of the 4th Conference of the European Society for Fuzzy Logic and Technology (Eusflat 2005) and 11th Rencontres Phrancophones sur la logique floue et ses applications (LFA’05)*, Barcelona, Spain, (2005) 1225–1230.
23. G. De Tré, and R. De Caluwe, “A Constraint Based Fuzzy Object Oriented Database Model”, in: Z. Ma (ed.), *Advances in Fuzzy Object-Oriented Databases: Modeling and Applications* (Idea Group Publishing, Hershey, USA, 2005) 1–45.
24. D. Dubois, and H. Prade, “Twofold fuzzy sets and rough sets – Some issues in knowledge representation”, *Fuzzy Sets and Systems* **23** (1987) 3–18.
25. D. Dubois, and H. Prade, *Possibility Theory* (Plenum Press, New York, USA, 1988).
26. D. Dubois, and H. Prade, “Possibility Theory, Probability Theory and Multiple-Valued Logics: A Clarification”, *Annals of Mathematics and Artificial Intelligence* **32** 1–4 (2001) 35–66.
27. D. Dubois, and H. Prade, “Bipolarity in flexible querying”, in: T. Andreasen et al. (eds.), *FQAS 2002*, LNAI 2522 (Springer-Verlag, Berlin, Heidelberg, Germany, 2002), 174–182.
28. B. de Finetti, *Theory of Probability* (Wiley, New York, USA, 1974).
29. R. George, A. Yazici, F.E. Petry, and B.P. Buckles, “Modeling Impreciseness and Uncertainty in the Object-Oriented Data Model – A Similarity-Based Approach”, in: R. De Caluwe (ed.), *Fuzzy and Uncertain Object-Oriented Databases: Concepts and Models* (World Scientific, Signapore, 1997), 63–95.
30. S. Gottwald, “Set theory for fuzzy sets of a higher level”, *Fuzzy Sets and Systems* **2** 2 (1979) 125–151.
31. A.M. Keller, and M. Winslett, “Approaches for Updating Databases With Incomplete Information and Nulls”, in: *Proc. of the 1st International Conference on Data Engineering*, Los Angeles, California, USA (1984) 332–340.
32. A.M. Keller, “Set-Theoretic Problems of Null Completion in Relational Databases”, *Information Processing Letters* **22** 5 (1986) 261–265.
33. Y. Kornatzky, and S.E. Shimony, “A Probabilistic Object-Oriented Data Model”, *Data and Knowledge Engineering* **12** 2 (1994) 143–166.
34. G. Kuper, L. Libkin, and J. Paredaens (eds.), *Constraint Databases* (Springer-Verlag, Berlin, Germany, 2000).
35. S.K. Lee, “An Extended Relational Database Model for Uncertain and Imprecise Information”, in: *Proc. of the 18th Conference on VLDB*, Vancouver, Canada (1992) 211–220.
36. D. Lucarella, G. Bordogna, and G. Pasi, “Pattern-Based Retrieval in a Fuzzy Object-Oriented Data Base”, in: *Proc. of the Tenth Annual ACM Symposium on Applied Computing, ACM-SAC’95*, Nashville, USA (1995) 508–513.
37. D. Lucarella, and A. Zanzi, “A Visual Retrieval Environment for Hypermedia Information Systems”, *ACM Transactions on Information Systems* **14** 1 (1996) 3–29.
38. Z. Ma (ed.), *Advances in Fuzzy Object-Oriented Databases: Modeling and Applications* (About Idea Group Inc, Hershey, USA, 2005).
39. A. Motro, “Management of Uncertainty in Database Systems”, in: W. Kim (ed.), *Modern Database Systems: The Object Model, Interoperability, and Beyond*, Addison-Wesley Publishing Company, Reading, Massachusetts, USA (1995) 457–476.

40. M. Oussalah, "Interval possibility measures: basic concepts and conditioning", *Kybernetes*, **32** 3 (2003) 317–342.
41. J. Paredaens, P. De Bra, M. Gyssens, and D. Van Gucht, *The Structure of the Relational Database Model* (Springer-Verlag, Berlin, Heidelberg, Germany, 1989).
42. G. Pasi, and R.R. Yager, "An approach to compute default attribute values in fuzzy object oriented data models", in: *Proc. of the FUZZ-IEEE, World Congress on Computational Intelligence,* Anchorage (1998) 1326–1331.
43. G. Pasi, and R.R. Yager, "Calculating attribute values using inheritance structures in fuzzy object-oriented data models", *IEEE Transactions on Systems, Man, and Cybernetics, Part C* **29** 4 (1999) 556–565.
44. F.E. Petry, *Fuzzy Databases: Principles and Applications* (Kluwer Academic Publishers, Boston, USA, 1996).
45. M. Pittarelli, "An Algebra for Probabilistic Databases", *IEEE Transactions on Knowledge and Data Engineering* **6** (1994) 293–303.
46. H. Prade, and C. Testemale, "Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries", *Information Sciences* **34** (1984) 115–143.
47. T. Radecki, "Mathematical Model of Information Retrieval Based on the Theory of Fuzzy Sets", in: *Communications of the Main Library and Scientific Information Center* **26D**, Technical University of Wroclaw, Poland (1975).
48. N. Rescher, *Many-Valued Logic* (Mc.Graw-Hill, New York, USA, 1969).
49. H. Riedel, and M.H. Scholl, "A Formalization of ODMG Queries", in: S. Spaccapietra en F. Maryanski (eds.), *Proc. of the 7th Working Conference on Database Semantics (DS-7)*, Leysin, Switzerland (1997) 63–90.
50. Y. Sagiv, "Can We Use the Universal Instance Assumption Without Using Nulls?", in: *Proc. of the SIGMOD Conference* (1981) 108–120.
51. C.A.B. Smith, "Consistency in Statistical Inference and Decision", *Journal of the Royal Statistical Society* **B23** (1961) 218–258.
52. J.D. Ullman, *Principles of Database Systems, 1st Edition* (Computer Science Press, 1980).
53. N. Van Gyseghem, *Een Vaag en Onzeker Objectgeoriënteerd Databankmodel: Studie, Ontwerp, Interface* (PhD thesis, Ghent University, 1995).
54. A. Van Schooten, *Ontwerp en implementatie van een model voor de representatie en manipulatie van onzekerheid en imprecisie in databanken en expertsystemen* (PhD thesis, Ghent University, 1988).
55. Y. Vassiliou, "Null Values in Data Base Management: A Denotational Semantics Approach", in: *Proc. of the SIGMOD Conference* (1979) 162–169.
56. P. Walley, *Statistical Reasoning with Imprecise Probabilities* (Chapman and Hall, London, England, 1991).
57. P. Walley, "Measures of uncertainty in expert systems", *Artificial Intelligence* **83** (1996) 1–58.
58. E. Wong, "A Statistical Approach to Incomplete Information in Database Systems", *ACM Transactions on Database Systems* **7** (1982) 470–488.
59. A. Yazici, and R. George, *Fuzzy Database Modeling* (Physica-Verlag, Heidelberg, Germany, 1999).
60. L.A. Zadeh, "Fuzzy Sets", *Information and Control* **8** 3 (1967) 338–353.
61. L.A. Zadeh, "Quantitative Fuzzy Semantics", *Information Sciences* **3** 2 (1971) 177–200.
62. L.A. Zadeh, "Fuzzy sets as a basis for a theory of possibility", *Fuzzy Sets and Systems* **1** 1 (1978) 3–28.
63. C. Zaniolo, "Database Relations with Null Values", *Journal of Computer and System Sciences* **28** 1 (1984) 142–166.

Chapter 5
Fuzzy querying of fuzzy databases

The fuzzy querying techniques for regular databases as presented in chapter 3 can be further extended (or generalized) so that they can be applied for the fuzzy querying of ‘fuzzy’ databases. The main difference with the ‘fuzzy’ querying techniques of regular databases is that in ‘fuzzy’ databases the data are not known with certainty in case of imprecision, fuzziness, uncertainty or missing information. This uncertainty propagates to, and should be reflected in, the query results so that the logical framework that is underlying the querying mechanism must be able to adequately model and deal with this uncertainty. In this chapter, fuzzy querying of fuzzy databases in three such frameworks is described. More specifically, attention is paid to querying in the *possibilistic framework*, in a *similarity based framework*, and in an *extended possibilistic framework*. The extended possibilistic framework can hereby be seen as an extension of the possibilistic framework.

For fuzzy querying of ‘fuzzy’ databases the same main ideas as with fuzzy querying of regular database remain applicable: preferences can be introduced *inside* elementary query conditions and *between* elementary query conditions [5]. For the treatment of preferences inside elementary querying conditions, the evaluation functions for simple conditions must be generalized in such a way that the possible imperfection of data is adequately dealt with. The treatment of composed conditions, taking into account preferences between querying conditions (if existent), can then occur completely analogously as with fuzzy querying of regular databases.

In the first section 5.1 of the chapter an example of a ‘fuzzy’ database is given. This example will be used in the remainder of this chapter to illustrate the presented techniques. The next sections deal with the possible logical frameworks supporting fuzzy querying of ‘fuzzy’ databases. In section 5.2 a general introduction is given. In section 5.3 the possibilistic approaches are described, section 5.4 deals with the similarity based approaches, and in section 5.5 the essence of the extended possibilistic approach is presented. For each of these approaches, attention is paid to the generalization of evaluation functions for simple search criteria. With the handling of the extended possibilistic approach, extra attention goes to the description of dealing with missing information in fuzzy query processing. The chapter ends with section 5.8 that deals with some relational and object oriented frameworks for fuzzy querying of 'fuzzy' databases.

5.1 Example database

The example database of section 3.1 can be adapted to a 'fuzzy' database as described below.

Example 5.1. Figure 5.1 contains a representation of the records of the adapted database. The database again consists of three record types 'Painting', 'Artist' and 'Owner'. To illustrate the handling of imperfect information the record types 'Painting' and 'Artist' have been generalized. The semantics of the record fields 'Artist', 'Period', 'Value' and 'Owner' of the record type 'Painting' and the semantics of the record fields 'Year_of_birth' and 'Year_of_death' of the record type 'Artist' are hereby generalized in such a way that these fields can contain possibility distributions as values (cf. the possibilistic relational model in section 4.2.1). These possibility distributions allow to model imprecision, uncertainty or unknown information. For the record fields 'Period', 'Value', 'Year_of_birth' and 'Year_of_death' all field values are represented by means of linguistic terms. Remark also that record fields that are used to associate record types with each other ('Artist' and 'Owner') can contain uncertain values. ◇

5.2 The evaluation of fuzzy query conditions

As already shortly explained in the introduction of this chapter, the evaluation of fuzzy querying conditions with 'fuzzy' databases has a lot of techniques in common with fuzzy query evaluation with regular databases. The differences stem from the fact that a 'fuzzy' database can contain imperfect data and these imperfections can bring along uncertainty with them.

Possible sources of imperfections are:

- With the integration of databases (e.g. when building a data warehouse) different values can be assigned to the same single-valued record field, where each of these values originates from a different data source.
- With predictions, one almost always has to deal with different possible/(un)certain data values, which eventually can be dependent of some parameters.
- In archives, information can (partially) get lost or (partially) get damaged. This can result in incomplete or missing data.
- Applying advanced analytical techniques, like for example clustering or pattern recognition, mostly results in different candidate objects or patterns.

RECORDTYPE Painting

ID	Name	Artist	Period	Value	Owner
P01	Fishermans house	{(Monet,1)}	1882	about_15M	{(Boijmans,1), (KMSK,1)}
P02	The ballet course	{(Degas,1)}	around_1870	more_than_8M	{(Louvre,1)}
P03	Mona Lisa	{(Da Vinci,1)}	very_old	very_expensive	{(Louvre,1)}
P04	Afternoon in Ostend	{(Ensor,1), (Permeke,0.4)}	1881	at_least_1K	{(KMSK,1)}

RECORDTYPE Artist

Name	First_name	Year_of_birth	Year_of_death
Da Vinci	Leonardo	around_1452	around_1519
Degas	Edgar	1834_or_1835	1917
Ensor	James	1860	1949
Monet	Claude	1840	1926
Permeke	Constant	1886	1952

RECORDTYPE Owner

Name	Place	Country
Boijmans	Rotterdam	The Netherlands
Louvre	Paris	France
KMSK	Antwerp	Belgium

Fig. 5.1 Records of the ‘fuzzy’ example database ‘Artworks’.

If the exact value of a record field is questioned and thus uncertain, the result of a query that acts on these field values can never be certain. As a consequence of this, the approaches with satisfaction degrees, presented in chapter 3 are no longer suited as an underlying logical framework. Indeed, these approaches do not allow it to model uncertainty. The other approaches, based on extended possibilistic truth values, can still be used as will be explained in what follows. Other, commonly used frameworks are the possibilistic framework and the probabilistic framework. The use of a similarity based framework has also been studied [6, 15]. Due to the scope of this book which is centralized around ‘fuzzy’ databases, only the possibilistic, the similarity based, and the extended possibilistic approaches are further described in this chapter.

5.3 Possibilistic approaches

5.3.1 Modelling

The possibilistic approaches are based on possibility theory [17, 11]. In these approaches the evaluation of the querying criteria results for each involved object in a **possibility** 'Pos' and a **necessity** 'Nec' which respectively denote to which extent the the object possibly satisfies the criteria and to which extent the object necessarily satisfies the criteria. With other words, if c is a querying condition and e denotes the evaluation function, then the evaluation $e(c)(r)$ of c for a database record r results in a couple

$$e(c)(r) = (\text{Pos}(c)(r), \text{Nec}(c)(r))$$

where $\text{Pos}(c)(r)$ is the possibility that r satisfies c and $\text{Nec}(c)(r)$ is the necessity that r satisfies c.

5.3.2 Evaluation of simple conditions

With fuzzy querying of 'fuzzy' databases two different kind of simple conditions exist:

- Conditions of the form

  $$A \ \theta \ L$$

  where A is a record field, L is a constant —possibly modelled by means of a fuzzy set— and θ represents a ('fuzzy') comparison operator or the compatibility operator.
- Conditions of the form

  $$A \ \theta \ B$$

  where A and B are record fields and θ represents a ('fuzzy') comparison operator.

As with fuzzy querying of regular databases, the comparison operators op, which might be 'fuzzy' or not, can be modelled by means of a membership function μ_{op} that is defined over the Cartesian product of two domains dom_1 and dom_2 and which denotes for each couple (v_1, v_2) of domain values $v_1 \in dom_1$ and $v_2 \in dom_2$ to which extent the operation $op(v_1, v_2)$ (or $v_1 \ op \ v_2$) is satisfied. Like with fuzzy querying of regular databases, this approach allows to deal with operators as for example 'approximately equal to' and 'much larger than'.

Simple conditions of the form $A \ \theta \ L$.

- **('Fuzzy') comparison operators (op).** With this first form of 'fuzzy' comparison operators, only one record field (or attribute) A is involved. Examples are

'Age is *much lower than* middle-aged' and 'Value is *approximately equal to* 3.000 Euro'.
If π_A represents the possibility distribution of the current field value of A in a record r of the 'fuzzy' database, μ_L is the membership function for the allowed values of A given by the user in the query specification and μ_{op} is the membership function of the ('fuzzy') comparison operator op which is defined over the Cartesian product $dom_A \times dom_A$ of the domain dom_A (of the data type of A) with itself, then the possibility measure and necessity measure of $A\,op\,L$ are obtained as:

$$e(A\,op\,L)(r) = (\text{Pos}(A\,op\,L)(r), \text{Nec}(A\,op\,L)(r))$$

where

– $\text{Pos}(A\,op\,L)(r) = \sup_{x \in dom_A} \min(\mu_{L\,\circ\,op}(x), \pi_A(x))$
– $\text{Nec}(A\,op\,L)(r) = \inf_{x \in dom_A} \max(\mu_{L\,\circ\,op}(x), 1 - \pi_A(x))$

with

$$\mu_{L\,\circ\,op}(x) = \sup_{x' \in dom_A} \min(\mu_{op}(x, x'), \mu_L(x')).$$

This last expression expresses that $L \circ op$ is the fuzzy set of the elements of dom_A which are in relation op with at least one element of L.
The possibility measure and necessity measure define two fuzzy sets over the result set of the comparison: the fuzzy set of records that possibly satisfy the comparison and the fuzzy set of records that necessarily satisfy the comparison.

- **Compatibility operator (*IS*).** As with the fuzzy querying of regular databases, the compatibility operator *IS* allows it to check to which extent the values of a given record field of database records are compatible with a fuzzy set of allowed values that is specified by the user in the query condition and is eventually represented by means of linguistic term. In 'fuzzy' databases, the record field values can be modelled by a possibility distributions. In general an 'IS'-proposition is of the form

$$A\,IS\,L$$

where A is a record field (or attribute) of the database and L is a fuzzy set of allowed values for A which is given by the user. L can eventually be represented by a linguistic term.
Examples of 'IS'-propositions are: '*Value* IS expensive', '*Speed* IS high', '*Period* IS old' and '*Weight* IS heavy'. Hereby, *Value*, *Speed*, *Period*, and *Weight* are record fields (or attributes) of record types stored in the database and 'expensive', 'high', 'old' and 'heavy' are linguistic terms which represent the domain values of the data type of the record field that by the user are considered as being allowed values. Each of these linguistic terms are modelled by a fuzzy set of which the membership grades are interpreted as degrees of compatibility.
From a possibilistic point of view, the evaluation of an 'IS'-proposition must now be interpreted as follows [13]: L is a fuzzy set and π_A is the possibility distribution that represents the field value of A. This possibility distribution models

an interpretation space where each interpretation corresponds to the assignment of one of the domain values x to the record field A with possibility $\pi_A(x)$. The evaluation of the 'IS'-proposition then corresponds to the determination of a possibility measure and necessity measure which together denote to which extent it is (un)certain that a given record satisfies the 'IS'-proposition.
Again, the possibility measure and necessity measure define two fuzzy sets over the result set of the compatibility operation: the fuzzy set of records that possibly satisfy the 'IS'-proposition and the fuzzy set of records that necessarily satisfy the 'IS'-proposition.
If π_A represents the possibility distribution of the field value of A which is obtained from a database record r and μ_L is the membership function given by the user that denotes the values of A that are considered to be adequate (or allowed) with respect to the query result, then the possibility measure and necessity measure of $A\ IS\ L$ are obtained as:

$$e(A\ IS\ L)(r) = (\text{Pos}(A\ IS\ L)(r), \text{Nec}(A\ IS\ L)(r))$$

where

– $\text{Pos}(A\ IS\ L)(r) = \sup_{x \in dom_A} \min(\pi_A(x), \mu_L(x))$
– $\text{Nec}(A\ IS\ L)(r) = \inf_{x \in dom_A} \max(\pi_A(x), \mu_L(x))$

Simple conditions of the form $A\ \theta\ B$.

With this form of simple conditions two different record fields (or attributes) A and B are involved. The operator θ represents a ('fuzzy') comparison operator which will further be denoted as op. In what follows it is for simplicity reasons assumed that A and B are independent attributes, i.e. the value of A is independent of the value of B and inversely, of the same record type. Examples of such conditions are 'the exam result for the course databases is *approximately equal to* the exam result for the course multimedia applications' and 'the exam result for the course databases is *much better than* the exam result for information management'.

If π_A and π_B represent the possibility distributions of the field values of A and B which are obtained from a database record r and μ_{op} is the membership function for the ('fuzzy') comparison operator op and is defined over the Cartesian product $dom_A \times dom_B$ of the domains dom_A and dom_B of the data types of respectively A and B, then the possibility measure and necessity measure of $A\ op\ B$ are obtained as:

$$e(A\ op\ B)(r) = (\text{Pos}(A\ op\ B)(r), \text{Nec}(A\ op\ B)(r))$$

where

- $\text{Pos}(A\ op\ B)(r) = \sup_{(x,x') \in dom_A \times dom_B} \min(\mu_{op}(x,x'), \pi_A(x), \pi_B(x'))$
- $\text{Nec}(A\ op\ B)(r) = \inf_{(x,x') \in dom_A \times dom_B} \max(\mu_{op}(x,x'), 1-\pi_A(x), 1-\pi_B(x'))$

As was previously the case, the possibility measure and necessity measure again define two fuzzy sets over the result set of the comparison: the fuzzy set of records that possibly satisfy the comparison and the fuzzy set of records that necessarily satisfy the comparison.

5.3.3 Evaluation of composite conditions

In case no preferences between query conditions are specified, composite conditions can be evaluated by means of the following computation rules:

- *Rule for negation:*
  If
  $$e(c) = (\text{Pos}(c), \text{Nec}(c))$$
  then
  $$e(\neg(c)) = (\overline{\text{Pos}(c)}, \overline{\text{Nec}(c)})$$
  where the line above the fuzzy set denotes the complement operator for fuzzy sets.
- *Rule for conjunction:*
  If
  $$e(c_1) = (\text{Pos}(c_1), \text{Nec}(c_1) \text{ and } e(c_2) = (\text{Pos}(c_2), \text{Nec}(c_2))$$
  then
  $$e(c_1 \wedge c_2)) = (\text{Pos}(c_1) \cap \text{Pos}(c_2), \text{Nec}(c_1) \cap \text{Nec}(c_2))$$
  where $\cap$ represents the intersection operator (t-norm) for fuzzy sets.
- *Rule for disjunction:*
  If
  $$e(c_1) = (\text{Pos}(c_1), \text{Nec}(c_1) \text{ and } e(c_2) = (\text{Pos}(c_2), \text{Nec}(c_2))$$
  then
  $$e(c_1 \vee c_2)) = (\text{Pos}(c_1) \cup \text{Pos}(c_2), \text{Nec}(c_1) \cup \text{Nec}(c_2))$$
  where $\cup$ represents the union operator (t-conorm) for fuzzy sets.

In case we have to deal with preferences between query conditions, an approach similar to the ones described in section 3.3.2.2 must be used [12].

5.4 Similarity relation based approaches

5.4.1 Modelling

In the similarity based approaches as introduced in section 4.2.2 the uncertainty involved in the query results is modelled by means of membership grades which are all interpreted as degrees of uncertainty [6, 15]. Thus, if r is a database object that is involved in the result of a query Q for which a querying condition c must be evaluated, then the evaluation

$$e(c)(r) = \mu_Q(r)$$

results in a membership degree $\mu_Q(r)$ that indicates how (un)certain it is that r satisfies the query Q.

5.4.2 Query evaluation

In the basic similarity relation based model, a general form of Boolean queries is considered [15]. A query

$$Q(a_1, a_2, \ldots, a_k),\ k \in \mathbb{N}$$

is hereby considered to be an expression of k factors $V_1, V_2, \ldots, V_k$ combined by disjunctive or conjunctive Boolean operators

$$V_1\ op\ V_2\ op\ \ldots\ op\ V_k.$$

In order to be well formed with respect to a (fuzzy) relation r having generalized domains $\tilde{dom}_{T_1}, \tilde{dom}_{T_2}, \ldots, \tilde{dom}_{T_m}$, $m \in \mathbb{N}$, each factor V_j, $1 \leq j \leq k$ must be

1. a domain element a, $a \in \tilde{dom}_{T_i}$, where $\tilde{dom}_{T_i}$ is a generalized domain for r, or
2. a domain element modified by one or more linguistic modifiers, like e.g. NOT, $VERY$ or $MORE\_OR\_LESS$.

The relation r may be one of the original database relations or be obtained as a result of fuzzy relational algebra operations. Fuzzy semantics apply to both operators and modifiers. An example of a query for a fuzzy relation ‘Painting’ with attributes *Value* and *Period* is

$$Q(cheap, recent) = MORE\_OR\_LESS\ cheap\ \text{ and }\ NOT\ VERY\ VERY\ recent$$

where ‘*cheap*’ is an abbreviation of the term ‘$Value = cheap$’ and ‘*recent*’ is an abbreviation of ‘$Period = recent$’. The use and interpretation of linguistic modifiers is as described in section 3.2.1.2. The linguistic modifier $VERY$ is hereby used to strengthen the linguistic term *recent*, NOT is used to negate ‘$VERY\ VERY\ recent$’, whereas $MORE\_OR\_LESS$ is used to dilate or to weaken the linguistic term ‘*cheap*’.

A membership grade is assigned with each tuple in a response relation r, reflecting the possibility that the tuple matches the query specifications. Let $a \in \tilde{dom}_{T_j}$ be an arbitrary element. The membership grade $\mu_a(b)$, $b \in \tilde{dom}_{T_j}$, is defined based on the similarity relation $S_{T_j}(a, b)$ over the domain $\tilde{dom}_{T_j}$. The query

$$Q(a_1, a_2, \ldots, a_k),\ k \in \mathbb{N}$$

induces a membership grade $\mu_Q(t)$ for a tuple t in the response r as follows:

1. Each interpretation $\alpha = [a'_1, a'_2, \ldots, a'_m]$ of t determines a grade $\mu_{a_j}(a'_j)$ for each domain element a_j of $Q(a_1, a_2, \ldots, a_k)$.
2. Evaluation of the modifiers and operators in $Q(a_1, a_2, \ldots, a_k)$ over the membership grades $\mu_{a_j}(a'_j)$ yields $\mu_Q(\alpha)$, the membership grade of the interpretation with respect to the query.
3. Finally, $\mu_Q(t) = \max\{\mu_Q(\alpha) | \alpha \text{ is an interpretation of } t\}$.

In short, the membership grade of a tuple represents the best matching interpretation. The response relation is then the set of tuples having non zero membership grades. In practice, it may be more realistic to consider only the tuple with the highest grade.

Three methods for calculating tuple membership grades are averaging, n-root and weighted summation of membership grades [15].

5.5 Extended possibilistic approach

5.5.1 Modelling

In the extended possibilistic approach, the framework based on (extended) possibilistic truth values ((E)PTVs) of chapter 3 is further developed. Uncertainty about the results of a fuzzy query is hereby modelled by means of (E)PTVs [8]. Consequently, an (E)PTV is associated with each object r that is involved in the result of a query for which a querying condition c must be evaluated. Thus,

$$e(c)(r) = \{(T, \mu_{e(c)(r)}(T)), (F, \mu_{e(c)(r)}(F)), (\perp, \mu_{e(c)(r)}(\perp))\}$$

The membership grades of the elements T, F and $\perp$ in this EPTV hereby respectively express to which extent it is possible that the object r satisfies condition c, to which extent it is not possible that the object r satisfies condition c, and to which extent it is possible that condition c is not applicable for the object r, i.e.

$$\begin{aligned} \mu_{e(c)(r)}(T) &= \text{Pos}(e(c)(r) = T) \\ \mu_{e(c)(r)}(F) &= \text{Pos}(e(c)(r) = F) \end{aligned}$$

and

$$\mu_{e(c)(r)}(\perp) = \text{Pos}(e(c)(r) = \perp).$$

The relationship between the possibilistic approach and the extended possibilistic approach follows from $\text{Nec}(c)(r) = 1 - \text{Pos}(\overline{c})(r)$, i.e.

- $\mu_{e(c)(r)}(T) = \text{Pos}(e(c)(r) = T) = \text{Pos}(c)(r)$
- $\mu_{e(c)(r)}(F) = \text{Pos}(e(c)(r) = F) = \text{Pos}(\overline{c})(r) = 1 - \text{Nec}(c)(r)$.

Differently than with the possibilistic approach, in the extended possibilistic approach the underlying logic allows it to explicitly reflect that some information might be inapplicable or non-existent: if for a given database record some of the querying conditions are not applicable, then this will be explicitly reflected in the resulting associated EPTV.

5.5.2 Evaluation of simple conditions

Also in the extended possibilistic approach a distinction is made between simple conditions of the form $A\ \theta\ L$ (where A is a record field, L is a constant —possibly modelled by a fuzzy set— and θ represents either a ('fuzzy') comparison operator or the compatibility operator) and simple conditions of the form $A\ \theta\ B$ (where A and B are record fields and θ represents a ('fuzzy') comparison operator).

The comparison operators op, fuzzy or not, are again modelled by means of a membership function μ_{op} which is defined over the Cartesian product of two domains dom_1 and dom_2 and which denote for each couple (v_1, v_2) of domain values $v_1 \in dom_1$ and $v_2 \in dom_2$ to which extent the operation $op(v_1, v_2$ (or $v_1\ op\ v_2$) is satisfied.

Simple conditions of the form $A\ \theta\ L$.

- **('Fuzzy') comparison operators (op).** If π_A is the possibility distribution of the current field value of A in a 'fuzzy' database record r, μ_L is the membership function for the allowed values for A specified by the user and μ_{op} is the membership function of the ('fuzzy') comparison operator op which is defined over the Cartesian product $dom_A \times dom_A$ of the domain dom_A (of the data type of A) with itself, then the (E)PTV of $A\ op\ L$ is obtained as:

  $$e(A\,op\,L)(r) = \{(T, \mu_{e(A\,op\,L)(r)}(T)), (F, \mu_{e(A\,op\,L)(r)}(F)), (\perp, \mu_{e(A\,op\,L)(r)}(\perp))\}$$

  where

  – $\mu_{e(A\,op\,L)(r)}(T) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\mu_{L\,\circ\,op}(x), \pi_A(x))$
  – $\mu_{e(A\,op\,L)(r)}(F) = 1 - \inf_{x \in dom_A \setminus \{\perp_{dom_A}\}} \max(\mu_{L\,\circ\,op}(x), 1 - \pi_A(x))$
  – $\mu_{e(A\,op\,L)(r)}(\perp) = \max(\mu_L(\perp_{dom_A}), \pi_A(\perp_{dom_A}))$

  with

  $$\mu_{L\,\circ\,op}(x) = \sup_{x' \in dom_A} \min(\mu_{op}(x, x'), \mu_L(x')).$$

  Hereby is is explicitly assumed that each domain dom_A contains a special domain value $\perp_{dom_A}$ which is used to model the inapplicability of a 'regular' domain value.
- **Compatibility operator (IS).** The compatibility operator IS allows it to compare a field value from the database —which is eventually modelled by a possibility distribution— with a fuzzy set of allowed values that specified by the user and

of which the membership grades are interpreted as degrees of compatibility. This fuzzy set can eventually be represented by means of a linguistic term. From a possibilistic point of view, the evaluation of an 'IS'-proposition can in the extended possibilistic approach be interpreted as follows [13]: L is a fuzzy set and π_A is the possibility distribution that represents the actual field value of A and models an interpretation space. Each interpretation corresponds to the assignment of one of the domain values x to the record field A. The possibility of the interpretation is $\pi_A(x)$. The evaluation of the 'IS'-proposition corresponds to the determination of a possibility measure and its complement which respectively denote to which extent it is possible that the stored field value is compatible with the given fuzzy set —represented by the membership grade of the truth value 'true' (T)— and to which extent it is not possible that the stored field value is compatible with the given fuzzy set —represented by the membership grade of the truth value 'false' (F)—. Additionally, it is also computed to which extent it is possible that the 'IS'-proposition is not applicable for the given record field.
If π_A represents the possibility distribution of the field value of A which is obtained from a 'fuzzy' database record r and μ_L is the membership function given by the user, denoting the values that considered to be adequate (or allowed) for A, then the (E)PTV of $A\ IS\ L$ is obtained as:

$$e(A\,IS\,L)(r) = \{(T, \mu_{e(A\,IS\,L)(r)}(T)), (F, \mu_{e(A\,IS\,L)(r)}(F)), (\perp, \mu_{e(A\,IS\,L)(r)}(\perp))\}$$

where

- $\mu_{e(A\,IS\,L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_A(x), \mu_L(x))$
- $\mu_{e(A\,IS\,L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_A(x), 1 - \mu_L(x))$
- $\mu_{e(A\,IS\,L)(r)}(\perp) = \min(\pi_A(\perp_{dom_A}), 1 - \mu_L(\perp_{dom_A}))$

Hereby it is again explicitly assumed that each domain dom_A contains a special domain value $\perp_{dom_A}$ that is used to denote the inapplicability or the non-existence of a 'regular' domain value.
The formula above reflects that:

- If A is possibly not applicable ($\pi_A(\perp_{dom_A}) > 0$) and the label L refers to the value $\perp_{dom_A}$ ($\mu_L(\perp_{dom_A}) > 0$), then the truth value T is possible to the extent that is calculated.
- The possibility of the truth value $\perp$ is 1 if A is possibly completely inapplicable ($\pi_A(\perp_{dom_A}) = 1$) and the label does not refer to the value $\perp_{dom_A}$.

Example 5.2. As an example, consider the evaluation of a proposition 'around_30K IS cheap' for a record field 'Value' as illustrated in figure 5.2. Hereby

- $\mu_{e(\text{'around\_30K IS cheap'})(r)}(T)$ is shortly denoted as μ_T,
- $\mu_{e(\text{'rond\_30 IS jong'})(r)}(F)$ is shortly denoted as μ_F, and
- $\mu_{e(\text{'rond\_30 IS jong'})(r)}(\perp) = 0$.

◇

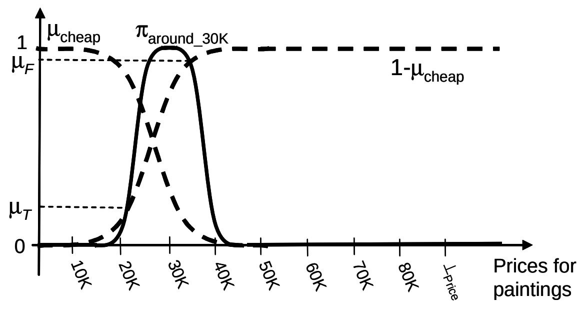

Fig. 5.2 Example of the evaluation of an 'IS'-proposition.

Simple conditions of the form $A\ \theta\ B$.

Two different record fields (or attributes) A and B are involved in these kinds of conditions. Furthermore, the operator θ represents a ('fuzzy') comparison which will further be denoted as op. For the sake of simplicity it is again assumed that A and B are independent attributes —the value of A is independent of the value of B and inversely— of the same record type.

If π_A and π_B represent the possibility distributions of thee field values of A and B which are obtained from a 'fuzzy' database record and μ_{op} is the membership function of the ('fuzzy') comparison operator op that is defined over the Cartesian product $dom_A \times dom_B$ of the domains dom_A and dom_B of the data types of respectively A and B, then the (E)PTV of $A\ op\ B$ is obtained as:

$$e(A\,op\,B)(r) = \{(T, \mu_{e(A\,op\,B)(r)}(T)), (F, \mu_{e(A\,op\,B)(r)}(F)), (\perp, \mu_{e(A\,op\,B)(r)}(\perp))\}$$

where

- $\mu_{e(A\,op\,B)(r)}(T) =$

$$\sup_{(x,x') \in dom_A \setminus \{\perp_{dom_A}\} \times dom_B \setminus \{\perp_{dom_B}\}} \min(\mu_{op}(x,x'), \pi_A(x), \pi_B(x'))$$

- $\mu_{e(A\,op\,B)(r)}(F) =$

$$1 - \inf_{(x,x') \in dom_A \setminus \{\perp_{dom_A}\} \times dom_B \setminus \{\perp_{dom_B}\}} \max(\mu_{op}(x,x'), 1 - \pi_A(x), 1 - \pi_B(x'))$$

- $\mu_{e(A\,op\,B)(r)}(\perp) = \max(\pi_A(\perp_{dom_A}), \pi_B(\perp_{dom_B}))$

Hereby it is explicitly assumed that the domain dom_A contains a special domain value $\perp_{dom_A}$ and the domain dom_B contains a special domain value $\perp_{dom_B}$ for the modelling of the inapplicability of a 'regular' domain value.

5.5.3 Evaluation of composite conditons

Because in the extended possibilistic approach all evaluations of simple conditions result in an EPTV the evaluation of composite conditions as can be done as has been explained in sections 3.2.2.2. In case preferences between query conditions have to be taken into account, the evaluation can be done as described in section 3.3.2.2.

5.6 Other approaches

As described in section 4.1.1 imprecise and vague information can in a ‘fuzzy’ database be modelled by means of ‘interval-valued’ fuzzy sets (IVFS), ‘intuitionistic’ fuzzy sets (IFS) or ‘two-fold’ fuzzy sets (TFS). With the handling of the extended possibilistic approach in section 5.5 it is already explained how ‘IS’-predicates of the form

$$A\ IS\ L$$

with A a record field (or attribute) of the ‘fuzzy’ database and L a fuzzy set of allowed values for A, can be evaluated. In remainder of this section, we further extend this approach in case the data are modelled by means of ‘interval-valued’, ‘intuitionistic’ or ‘two-fold’ fuzzy sets (cf. [9]).

5.6.1 Evaluation of the compatibility operator ‘IS’ with the use of ‘intuitionistic’ fuzzy sets

First we deal with the case based on ‘intuitionistic’ fuzzy sets (IFS) because this offers interesting extra querying facilities. At the one hand it is assumed that the values of the record field A of an ‘IS’-proposition are allowed to be modelled by an ‘intuitionistic’ possibility distribution (IPD)

$$(\pi_{\mu_A}, \pi_{\nu_A}).$$

At the other hand the linguistic term L can also be modelled by an ‘intuitionistic’ fuzzy set (IFS)

$$L = \{< x, \mu_L(x), \nu_L(x) > | x \in dom_A\}$$

with membership function μ_L and non-membership function ν_L.

Case 1

In the case where the value of A that is obtained from the ‘fuzzy’ database record r is modelled by a regular possibility distribution

$$\pi_A$$

and the linguistic term L is given by means of an intuitionistic fuzzy set

$$\{< x, \mu_L(x), \nu_L(x) > | x \in dom_A\}$$

the (E)PTV of the proposition

$$A\ IS\ L$$

is defined by

$$e(A\ IS\ L)(r) = \{(T, \mu_{e(A\ IS\ L)(r)}(T)), (F, \mu_{e(A\ IS\ L)(r)}(F)), (\perp \mu_{e(A\ IS\ L)(r)}(\perp))\}$$

where

- $\mu_{e(A\ IS\ L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_A(x), \mu_L(x))$
- $\mu_{e(A\ IS\ L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_A(x), \nu_L(x))$
- $\mu_{e(A\ IS\ L)(r)}(\perp) = \min(\pi_A(\perp_{dom_A}), \nu_L(\perp_{dom_A}))$

Example 5.3. The same situation as in example 5.2, considering the evaluation of the proposition 'around_30K IS cheap' for a record field 'Value', is given in figure 5.3. The linguistic term 'cheap' is hereby now modelled by an IFS. ◇

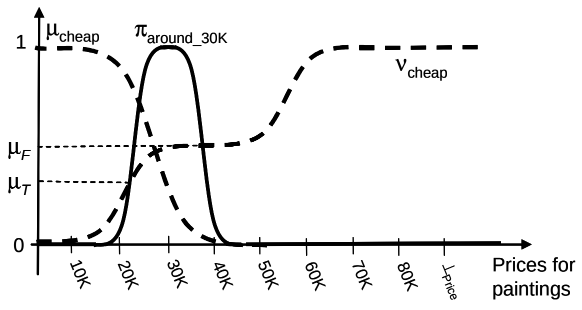

Fig. 5.3 Example of the evaluation of an 'IS'-proposition with L being an IFS.

By using an IFS, information about the non-membership of elements that are denoted by the linguistic term L is now more generally modelled by the non-membership function ν_L, instead of by the complement $1 - \mu_L$. This allows for example, as illustrated in figure 5.3, to consider prices between 30K and 50K as being neither cheap, nor not cheap. This illustrates that 'intuitionistic' fuzzy sets allow for more flexibility in expressing non-membership and can be used in (database) applications which demand for such a flexibility.

Case 2

In the case where the value of A that is obtained from the ‘fuzzy’ database record r is modelled by an ‘intuitionistic’ possibility distribution (IPD)

$$(\pi_{\mu_A}, \pi_{\nu_A})$$

and the linguistic term L is given by an ‘intuitionistic’ fuzzy set (IFS)

$$\{< x, \mu_L(x), \nu_L(x) > | x \in dom_A\}$$

the (E)PTV of the proposition

$$A\ IS\ L$$

is analogously defined by

$$e(A\ IS\ L)(r) = \{(T, \mu_{e(A\ IS\ L)(r)}(T)), (F, \mu_{e(A\ IS\ L)(r)}(F)), (\perp, \mu_{e(A\ IS\ L)(r)}(\perp))\}$$

where

- $\mu_{e(A\ IS\ L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_{\mu_A}(x), \mu_L(x))$
- $\mu_{e(A\ IS\ L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_{\mu_A}(x), \nu_L(x))$
- $\mu_{e(A\ IS\ L)(r)}(\perp) = \min(\pi_{\mu_A}(\perp_{dom_A}), \nu_L(\perp_{dom_A}))$

The non-membership part π_{ν_A} of the IPD $(\pi_{\mu_A}, \pi_{\nu_A})$ for A is not used for the computation of the (E)PTV $e(A\ IS\ L)(r)$. This is due to the fact that we are only interested in the compatibility between A and L (and $NOT(L)$) and not in the compatibility between $NOT(A)$ and L (and $NOT(L)$). In cases where we have to compute the (E)PTV of a proposition of the form

$$NOT(A)\ IS\ L$$

the part π_{ν_A} can be meaningfully used as follows:

$$e(NOT(A)\ IS\ L)(r) = \{(T, \mu_{e(NOT(A)\ IS\ L)(r)}(T)),$$
$$(F, \mu_{e(NOT(A)\ IS\ L)(r)}(F)), (\perp, \mu_{e(NOT(A)\ IS\ L)(r)}(\perp))\}$$

where

- $\mu_{e(NOT(A)\ IS\ L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_{\nu_A}(x), \mu_L(x))$
- $\mu_{e(NOT(A)\ IS\ L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_{\nu_A}(x), \nu_L(x))$
- $\mu_{e(NOT(A)\ IS\ L)(r)}(\perp) = \min(\pi_{\nu_A}(\perp_{dom_A}), \nu_L(\perp_{dom_A}))$

5.6.2 *Evaluation of the compatibility operator ‘IS’ with the use of ‘interval-valued’ fuzzy sets*

On the one hand, ‘interval-valued’ fuzzy sets can be used to model the values of a record field A in a ‘fuzzy’ database. This can be done by means of an interval-valued possibility distribution (IVPD)

$$(\pi_A^l, \pi_A^u).$$

In such a case, the record field A in an ‘IS’-proposition

$$A\ IS\ L$$

can also take an IVPD as value. On the other hand the linguistic term L in the ‘IS’-proposition can be modelled by means of an interval-valued fuzzy set (IVFS)

$$L = \{< x, \mu_L^l(x), \mu_L^u(x) > | x \in dom_A\}$$

with lower bound μ_L^l and upper bound μ_L^u for the membership function. Hereby, μ_L^u can be seen as an optimistic approximation of the membership function, whereas μ_L^l can be seen as a pessimistic approximation.

Case 1

We first consider the case where the value of the record field A obtained from the ‘fuzzy’ database record r is modelled by a regular possibility distribution

$$\pi_A$$

and the linguistic term L is given by means of an IVFS

$$L = \{< x, \mu_L^l(x), \mu_L^u(x) > | x \in dom_A\}.$$

In such a case and if we consider an optimistic approach, the (E)PTV of the proposition

$$A\ IS\ L$$

is defined by

$$e(A\ IS\ L)(r) = \{(T, \mu_{e(A\ IS\ L)(r)}(T)), (F, \mu_{e(A\ IS\ L)(r)}(F)), (\perp, \mu_{e(A\ IS\ L)(r)}(\perp))\}$$

where

- $\mu_{e(A\ IS\ L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_A(x), \mu_L^u(x))$
- $\mu_{e(A\ IS\ L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_A(x), 1 - \mu_L^u(x))$
- $\mu_{e(A\ IS\ L)(r)}(\perp) = \min(\pi_A(\perp_{dom_A}), 1 - \mu_L^u(\perp_{dom_A}))$

Example 5.4. Let us, consider the same proposition ‘around_30K IS cheap’ as in examples 5.2 and 5.3. If the linguistic term ‘cheap’ is modelled by an IVFS as depicted in figure 5.2, the computation of the resulting (E)PTV is done as illustrated in the figure. ◇

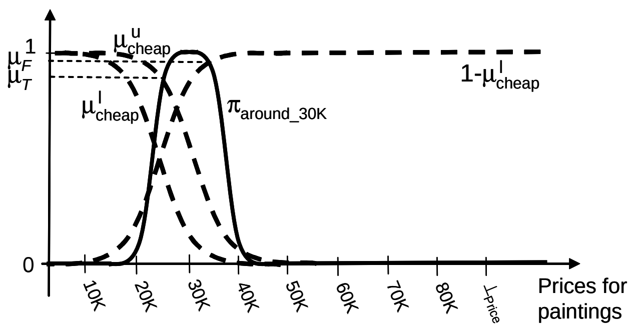

Fig. 5.4 Example of the evaluation of an ‘IS’-proposition with L an IVFS.

To determine the (E)PTV the (optimistic) upper bound μ_L^u is used to compute the possibility of the truth value T, while the (optimistic) complement of lower bound μ_L^u is used to compute the possibilities of the truth values F and $\perp$. This allows for more flexibility because imprecision in the modelling of linguistic terms can be more adequately dealt with.

In the case of a pessimistic approach, the (E)PTV of the proposition

$$A\ IS\ L$$

is defined by

$$e(A\ IS\ L)(r) = \{(T, \mu_{e(A\ IS\ L)(r)}(T)), (F, \mu_{e(A\ IS\ L)(r)}(F)), (\perp, \mu_{e(A\ IS\ L)(r)}(\perp))\}$$

where

- $\mu_{e(A\ IS\ L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_A(x), \mu_L^l(x))$
- $\mu_{e(A\ IS\ L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_A(x), 1 - \mu_L^l(x))$
- $\mu_{e(A\ IS\ L)(r)}(\perp) = \min(\pi_A(\perp_{dom_A}), 1 - \mu_L^l(\perp_{dom_A}))$

Case 2

Secondly we consider the case where the value of A that is obtained from the ‘fuzzy’ database record r is modelled by an interval-valued possibility distribution

$$(\pi_A^l, \pi_A^u)$$

and the linguistic term L is given by means of an IVFS

$$L = \{< x, \mu_L^l(x), \mu_L^u(x) > | x \in dom_A\}.$$

Also in such a case, either an optimistic or a pessimistic approach is possible for the calculation of the (E)PTV of the proposition

$$A\,IS\,L.$$

With the pessimistic approach the (E)PTV is defined by

$$e(A\,IS\,L)(r) = \{(T, \mu_{e(A\,IS\,L)(r)}(T)), (F, \mu_{e(A\,IS\,L)(r)}(F)), (\perp, \mu_{e(A\,IS\,L)(r)}(\perp))\}$$

where

- $\mu_{e(A\,IS\,L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_A^l(x), \mu_L^l(x))$
- $\mu_{e(A\,IS\,L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_A^l(x), 1 - \mu_L^l(x))$
- $\mu_{e(A\,IS\,L)(r)}(\perp) = \min(\pi_A^l(\perp_{dom_A}), 1 - \mu_L^l(\perp_{dom_A}))$

Hereby, the (pessimistic) lower bound function π_A^l is used.

Whereas with the optimistic approach the (E)PTV is defined by

$$e(A\,IS\,L)(r) = \{(T, \mu_{e(A\,IS\,L)(r)}(T)), (F, \mu_{e(A\,IS\,L)(r)}(F)), (\perp, \mu_{e(A\,IS\,L)(r)}(\perp))\}$$

where

- $\mu_{e(A\,IS\,L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_A^u(x), \mu_L^u(x))$
- $\mu_{e(A\,IS\,L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_A^u(x), 1 - \mu_L^u(x))$
- $\mu_{e(A\,IS\,L)(r)}(\perp) = \min(\pi_A^u(\perp_{dom_A}), 1 - \mu_L^u(\perp_{dom_A}))$

Now, the (optimistic) upper bound function π_A^u is used.

5.6.3 *Evaluation of the compatibility operator ‘IS’ with the use of ‘two-fold’ fuzzy sets*

If ‘two-fold’ fuzzy sets are considered, then it is an option to model the values of the record field A of an ‘IS’-proposition by means of a ‘two-fold’ possibility distribution (TPD)

$$(\pi_A^P, \pi_A^S).$$

Furthermore, it is also an option to model the linguistic term L by means of a ‘two-fold’ fuzzy set (TFS)

$$L = (\{< x, \mu_L^P(x) > | x \in dom_A\}, \{< x, \mu_L^S(x) > | x \in dom_A\})$$

with membership function μ_L^P for the determination of the preferred values and membership function μ_L^S for the determination of the satisfactory values. Hereby,

μ_L^P can be seen as the preferred conformity for the membership function of L, while μ_L^S can be interpreted as an allowed conformity for the membership function of L.

Case 1

In the case where the value of the record field A obtained from the 'fuzzy' database record r is modelled by a regular possibility distribution

$$\pi_A$$

and the linguistic term L is given by means of a TFS

$$L = (\{< x, \mu_L^P(x) > | x \in dom_A\}, \{< x, \mu_L^S(x) > | x \in dom_A\})$$

both membership functions μ_L^P and μ_L^S can be used to compute the (E)PTV of the proposition

$$A \; IS \; L.$$

With a progressive, optimistic approach, the (E)PTV can be defined on the basis of the membership function μ_L^S that defines the allowed values, i.e.:

$$e(A \; IS \; L)(r) = \{(T, \mu_{e(A \; IS \; L)(r)}(T)), (F, \mu_{e(A \; IS \; L)(r)}(F)), (\perp, \mu_{e(A \; IS \; L)(r)}(\perp))\}$$

where

- $\mu_{e(A \; IS \; L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_A(x), \mu_L^S(x))$
- $\mu_{e(A \; IS \; L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_A(x), 1 - \mu_L^S(x))$
- $\mu_{e(A \; IS \; L)(r)}(\perp) = \min(\pi_A(\perp_{dom_A}), 1 - \mu_L^S(\perp_{dom_A}))$

With a rather conservative, pessimistic approach, the definition of the (E)PTV can be based on the membership function μ_L^P that defines the preferred values, i.e.:

$$e(A \; IS \; L)(r) = \{(T, \mu_{e(A \; IS \; L)(r)}(T)), (F, \mu_{e(A \; IS \; L)(r)}(F)), (\perp, \mu_{e(A \; IS \; L)(r)}(\perp))\}$$

where

- $\mu_{e(A \; IS \; L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_A(x), \mu_L^P(x))$
- $\mu_{e(A \; IS \; L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_A(x), 1 - \mu_L^P(x))$
- $\mu_{e(A \; IS \; L)(r)}(\perp) = \min(\pi_A(\perp_{dom_A}), 1 - \mu_L^P(\perp_{dom_A}))$

Example 5.5. In figure 5.5 the computation of the (E)PTV resulting from the evaluation of the proposition 'around_30K IS cheap' is illustrated. For the sake of the example we have chosen for the conservative approach. The stored field value 'around_30K' is modelled by a regular possibility distribution, whereas the linguistic term 'cheap' is modelled by the TFS with membership functions μ_{cheap}^P and μ_{cheap}^S. $\diamond$

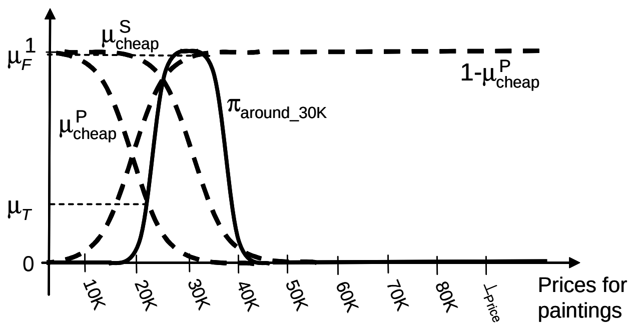

Fig. 5.5 Example of the evaluation of an 'IS'-proposition with L an IFS (conservative approach).

Case 2

In the case where the value of the record field A obtained from the 'fuzzy' database record r is modelled by a 'two-fold' possibility distribution

$$(\pi^P_A, \pi^S_A)$$

and the linguistic term L is given by means of a TFS

$$L = (\{< x, \mu^P_L(x) > | x \in dom_A\}, \{< x, \mu^S_L(x) > | x \in dom_A\})$$

we can again choose for an optimistic or a pessimistic approach for the computation of the (E)PTV of the proposition

$$A\; IS\; L.$$

With a progressive, optimistic approach, the (E)PTV can be defined on the basis of the membership functions π^S_A and μ^S_L that depart from the allowed values, i.e.:

$$e(A\; IS\; L)(r) = \{(T, \mu_{e(A\; IS\; L)(r)}(T)), (F, \mu_{e(A\; IS\; L)(r)}(F)), (\perp, \mu_{e(A\; IS\; L)(r)}(\perp))\}$$

where

- $\mu_{e(A\; IS\; L)(r)}(T) = \sup_{x \in dom_A} \min(\pi^S_A(x), \mu^S_L(x))$
- $\mu_{e(A\; IS\; L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi^S_A(x), 1 - \mu^S_L(x))$
- $\mu_{e(A\; IS\; L)(r)}(\perp) = \min(\pi^S_A(\perp_{dom_A}), 1 - \mu^S_L(\perp_{dom_A}))$

With a conservative, pessimistic approach, the definition of the (E)PTV can be based on the membership functions π^P_A and μ^P_L that depart from the preferred values, i.e.:

$$e(A\; IS\; L)(r) = \{(T, \mu_{e(A\; IS\; L)(r)}(T)), (F, \mu_{e(A\; IS\; L)(r)}(F)), (\perp, \mu_{e(A\; IS\; L)(r)}(\perp))\}$$

where

- $\mu_{e(A\,IS\,L)(r)}(T) = \sup_{x \in dom_A} \min(\pi_A^P(x), \mu_L^P(x))$
- $\mu_{e(A\,IS\,L)(r)}(F) = \sup_{x \in dom_A \setminus \{\perp_{dom_A}\}} \min(\pi_A^P(x), 1 - \mu_L^P(x))$
- $\mu_{e(A\,IS\,L)(r)}(\perp) = \min(\pi_A^P(\perp_{dom_A}), 1 - \mu_L^P(\perp_{dom_A}))$

5.7 'Fuzzy' database querying in the presence of null values

As explained in section 4.1.3, the extended possibilistic approach provides extra facilities for the handling of missing information. In this subsection attention is paid to the different cases and contexts where a null value can occur with the handling of selection conditions of flexible queries. The selection operator is the key operator to deal with when handling null values. The handling of other operators, like e.g. projection and join —which is typical for relational databases—, is not further dealt with in this section.

For the sake of illustration, a 'fuzzy' database that consists of one single record type, called '*Poll*', is considered. With this record type (partial) information resulting from a social neighbourhood research is stored in the database. Each record r of the record type represents information about a registered participant of the research and is characterized by a unique participant identifier (PID) —the key field—, a field for the year of birth of the participant ($Year\_of\_birth$), a salary field ($Salary$) and a field with the year of birth of the oldest child of the participant ($Oldest\_child$). As extra information a field ($EPTV$) is added to represent the EPTV that is used to express the extent to which it is (un)certain that the record r satisfies the predicate of the record type (cf. subsection 3.4.1.2). For the sake of simplicity it is assumed that all records initially have been assigned an EPTV $\{(T,1)\}$. The value $\{(T,1)\}$ can for example be the default value that is assigned with a record on insertion. Alternatively, in a more general situation the user could be allowed to provide an extended possibilistic truth value to express that the record can not be considered as fully satisfying (or 'belonging to') the record type.

In what follows it is also assumed for the sake of simplicity that the non-key record fields are mutually independent of each other —the value of a record field is independent of the values of the other record fields and inversely—. Without this restriction, the following description would be more complex because dependencies between record fields must be taking into account when handling missing information: if a value is missing, but its dependent values are not, then the missing value can in some cases be (approximately)derived from its dependent values. For example, if the salary of a person is dependent of his or her age, then a missing salary value could be derived from the known year of birth of the person.

For the modelling of undefined information, the domains dom_t of the associated data types t of the record fields are considered to contain a type specific domain value $\perp_t$. The value $\perp_t$ is used to model those cases where a regular domain value of t does not apply. Furthermore, for each domain dom_t three linguistic terms 'UNK', 'UNA' and 'N/A' are defined as described in subsection 4.1.3.2.

PID	$Year\_of\_birth$	$Salary$	$Oldest\_child$	$EPTV$
P01	1969	2.600	2000	$\{(T,1)\}$
P02	1952	4.000±1000	1976	$\{(T,1)\}$
P03	1966	2.800	UNK	$\{(T,1)\}$
P04	1930	N/A	1960 or 1961	$\{(T,1)\}$
P05	1965	3.000±500	N/A	$\{(T,1)\}$
P06	1955	‘not_average’	N/A	$\{(T,1)\}$
P07	1962	UNK	UNK	$\{(T,1)\}$
P08	1958	UNK	N/A	$\{(T,1)\}$
P09	1930	N/A	N/A	$\{(T,1)\}$
P10	1980	UNA	UNA	$\{(T,1)\}$

Table 5.1 Examples of records of the record type *Poll*.

Table 5.1 contains a table representation of the record set of the record type *Poll* (each row represents a record). The values of the ‘*Salary*’ and ‘*Oldest_child*’ fields are all labels that are modelled by a possibility distribution that is defined over the domain of the data type of the record field. A label with a regular number like ‘2.600’ corresponds with a possibility distribution that is characterized by normalized membership function of which the support is a singleton. Consequently, labels like ‘1960 or 1961’ correspond with a possibility distribution which is characterized by a normalized membership function with a discrete support. Labels like ‘4.000±1.000’ correspond with possibility distributions with triangular membership functions. As such, ‘4.000±1.000’ is for example modelled by the triangular distribution function

$$\pi_{4.000\pm1.000}(x) = \begin{cases} \frac{x-3000}{1000} & \text{iff } x \in [3000,4000] \\ \frac{5000-x}{1000} & \text{iff } x \in [4000,5000] \\ 0 & \text{else} \end{cases}$$

Analoguously, the possibility distributions that correspond with the labels ‘average’ and ‘not_average’ are respectively defined by the trapezoidal distribution function

$$\pi_{average}(x) = \begin{cases} \frac{x-1000}{500} & \text{iff } x \in [1000,1500] \\ 1 & \text{als } x \in [1500,2500] \\ \frac{3000-x}{500} & \text{iff } x \in [2500,3000] \\ 0 & \text{else} \end{cases}$$

and the distribution function

$$\pi_{not\_average}(x) = 1 - \pi_{average}(x)$$

Hereby the salary value of record ‘P06’ is either ‘lower’ than 1.500, or ‘larger’ than 2.500, or ‘not applicable’ which can for example be due to the fact that participant ‘P06’ is not an employee or is retired.

PID	*Salary*	*EPTV*
P01	2.600	$\{(T,1)\}$
P04	N/A	$\{(\perp,1)\}$
P05	3.000±500	$\{(T,0.2),(F,1)\}$
P06	'not_average'	$\{(T,0.2),(F,1),(\perp,1)\}$
P07	UNK	$\{(T,1),(F,1)\}$
P08	UNK	$\{(T,1),(F,1)\}$
P09	N/A	$\{(\perp,1)\}$
P10	UNA	$\{(T,1),(F,1),(\perp,1)\}$

Table 5.2 Result of Query 1.

To illustrate database querying, a simple SELECT-FROM-WHERE grammar will be used. In this grammar the selection conditions are extended with 'IS'-propositions which are evaluated as explained in section 5.5. As an example of a very straightforward query with a simple selection condition the following query specification can be considered :

Query 1

SELECT *PID*, *Salary* FROM *Poll* WHERE *Salary* IS 2.600

This query selects the participant identifier (*PID*) and salary (*Salary*) of all participants that have a salary that exactly equals 2.600. Suppose that the query is executed on the database presented in table 5.1. The records that then belong to the result of Query 1 are given in table 5.2. For each record in the result, the corresponding EPTV is computed as described in section 5.5. (Records with an associated EPTV $\{(F,1)\}$ are omitted.) As with regular database querying, the extended possibilistic approach allows it to find records for which the corresponding truth value is either completely true (record 'P01'), or completely false (records 'P02' and 'P03'). Additionally, the approach also allows it to deal with records that only partially satisfy the selection condition (records 'P05' and 'P06'). Cases of which it is known that a regular salary value exists, but for which there is further nothing known about this salary value result in an EPTV representing 'unknown (but applicable)' (records 'P07' and 'P08'). If it is known that a regular salary value does not apply for a given participant, because this participant is for example not an employee or retired, then the corresponding EPTV represents 'not applicable' (records 'P04' and 'P09'). The possibility of inapplicability can also occur in combination with the possibility of a partial query satisfaction (record 'P06'). Finally, it can occur that absolutely nothing is known about the salary value of a given record. In such a case, the corresponding EPTV represents 'not available' (record 'P10').

In a more complex query, the selection criteria can contain labels which can not be modelled by means of a regular set. For example, consider the next query with a simple fuzzy selection condition:

PID	*Salary*	*EPTV*
P01	2.600	$\{(T,0.8),(F,0.2)\}$
P03	2.800	$\{(T,0.4),(F,0.6)\}$
P04	N/A	$\{(\perp,1)\}$
P05	3.000±500	$\{(T,0.5),(F,1)\}$
P06	'not_average'	$\{(T,0.5),(F,1),(\perp,1)\}$
P07	UNK	$\{(T,1),(F,1)\}$
P08	UNK	$\{(T,1),(F,1)\}$
P09	N/A	$\{(\perp,1)\}$
P10	UNA	$\{(T,1),(F,1),(\perp,1)\}$

Table 5.3 Result of Query 2.

Query 2

SELECT *PID*, *Salary* FROM *Poll* WHERE *Salary* IS *average*

This query selects the participant identifier (*PID*) and salary (*Salary*) of all participants that have an average salary. Hereby it is assumed that 'average' is a linguistic term that represents $\pm[1500–2500]$ and is modelled by the trapezoidal distribution function given above in the description of table 5.1. The records that belong to the result if the query is executed on the database presented in table 5.1 are given in table 5.3. This result illustrates that EPTVs can meaningfully be used to express the uncertainty about query satisfaction and moreover allow to adequately handle missing information with fuzzy database querying. More specifically, EPTVs allow to model those cases where the satisfaction of a selection condition is not completely certain (records 'P01', 'P03', 'P05' and 'P06'). It can also occur that the selection condition is completely not satisfied (record 'P02'). As is the case with regular queries, EPTVs also allow in fuzzy querying to adequately deal with unknown information (records 'P07' and 'P08'), the (possible) inapplicability of information (records 'P04', 'P06' and 'P09') and the unavailability of information (record 'P10').

With composite querying conditions the EPTVs in the result of a query are computed from the EPTVs of the simple conditions of the composition. Hereby, the logical operators '$\tilde{\neg}$', '$\tilde{\wedge}$' and '$\tilde{\vee}$', which are defined in subsection 2.4.2.2 can for example be used. For the sake of illustration, we can consider we can consider the following the query:

Query 3

SELECT *PID*, *Salary*, *Oldest_child* FROM *Poll*
WHERE (*Salary* IS 2.600) AND (*Oldest_child* IS 2000)

This query selects the participant identifier (*PID*), salary (*Salary*), and year of birth of the oldest child (*Oldest_child*) of all participants that have a salary that is exactly equal to 2.600 and of who the oldest child is born in the year 2000. The

PID	$e(c_1)(r)$	$e(c_2)(r)$	$e(c_1)(r)\tilde{\wedge}e(c_2)(r)$
P01	$\{(T,1)\}$	$\{(T,1)\}$	$\{(T,1)\}$
P02	$\{(F,1)\}$	$\{(F,1)\}$	$\{(F,1)\}$
P03	$\{(F,1)\}$	$\{(T,1),(F,1)\}$	$\{(F,1)\}$
P04	$\{(\perp,1)\}$	$\{(F,1)\}$	$\{(F,1)\}$
P05	$\{(T,0.2),(F,1)\}$	$\{(\perp,1)\}$	$\{(F,1),(\perp,0.2)\}$
P06	$\{(T,0.2),(F,1),(\perp,1)\}$	$\{(\perp,1)\}$	$\{(F,1),(\perp,1)\}$
P07	$\{(T,1),(F,1)\}$	$\{(T,1),(F,1)\}$	$\{(T,1),(F,1)\}$
P08	$\{(T,1),(F,1)\}$	$\{(\perp,1)\}$	$\{(F,1),(\perp,1)\}$
P09	$\{(\perp,1)\}$	$\{(\perp,1)\}$	$\{(\perp,1)\}$
P10	$\{(T,1),(F,1),(\perp,1)\}$	$\{(T,1),(F,1),(\perp,1)\}$	$\{(T,1),(F,1),(\perp,1)\}$

Table 5.4 Calculation of the EPTVs from the result of Query 3.

PID	$Salary$	$Oldest\_Child$	$EPTV$
P01	2.600	2000	$\{(T,1)\}$
P05	3.000±500	N/A	$\{(F,1),(\perp,0.2)\}$
P06	'not_average'	N/A	$\{(F,1),(\perp,1)\}$
P07	UNK	UNK	$\{(T,1),(F,1)\}$
P08	UNK	N/A	$\{(F,1),(\perp,1)\}$
P09	N/A	N/A	$\{(\perp,1)\}$
P10	UNA	UNA	$\{(T,1),(F,1),(\perp,1)\}$

Table 5.5 Result of Query 3.

result of query 3, in case the query is executed on the example database presented in table 5.1, is given in table 5.5. For each record of the result, the associated EPTV is computed by applying the conjunction operator $\tilde{\wedge}$ on the EPTVs that result from the evaluation of the respective criteria $c_1 =$'Salary IS 2.600' and $c_2 =$'Oldest_Child IS 2000', as presented in table 5.4. In general, the use of extended possibilistic logic —based on the operators $\tilde{\wedge}$, $\tilde{\vee}$ and $\tilde{\neg}$— allows it to adequately handle missing information in the case of composite querying conditions.

Records for which both criteria are completely certainly satisfied (record 'P01') certainly belong to the query result, whereas records for which both criteria are completely certainly not satisfied (record 'P02') certainly do not belong to the query result. If one of the criteria completely certainly is not satisfied, then the record certainly does not belong to the query result, independent of the result of the evaluation of the other criterion (records 'P02', 'P03' and 'P04'). The conjunction of a criterion that is certainly satisfied and a criterion that is certainly not applicable results in a truth value that is possibly undefined (records 'P05' and 'P06'). Cases where both criteria either evaluate to an EPTV that represents 'unknown' (record 'P07') or an EPTV that represents 'not applicable' (record 'P09'), respectively result in an EPTV 'unknown' and 'not applicable'. If nothing is known about the satisfaction of both criteria, then the resulting EPTV represents 'not available' (record 'P10').

PID	$Salary$	$EPTV$
P04	N/A	$\{(T,1)\}$
P06	'not_average'	$\{(T,1),(F,1)\}$
P09	N/A	$\{(T,1)\}$
P10	UNA	$\{(T,1),(F,1)\}$

Table 5.6 Result of Query 4.

The conjunction of an EPTV that represents 'unknown' and an EPTV that represents 'not applicable' results in an EPTV that represents 'completely possible false or completely possible not applicable' (record 'P08').

To end this section, we still consider three special types of queries where the database is respectively queried for data that are compatible with the predefined labels 'N/A', 'UNK' and 'UNA'. These queries respectively search for data i) where the attribute under consideration is not applicable, ii) where the attribute under consideration is applicable, but there are no further restrictions imposed on the data, and iii) where the attribute under consideration is either not applicable, or applicable with no further restrictions on the data. Such queries —especially the last one— are clearly not very useful in practice, but are considered here to clarify the behaviour of the presented approach.

Query 4

SELECT *PID*, *Salary* FROM *Poll* WHERE *Salary* IS N/A

With query 4, the database is explicitly queried for records in which a regular salary value is not provided. This means that '*Salary* IS N/A' must be either interpreted as 'possibly no salary value existent' or 'possibly no regular salary value applicable'. The result of this query, if executed on the example database presented in table 5.1, is given in table 5.6. Records with a salary value 'N/A' completely certainly satisfy the selection condition (records 'P04' and 'P09'). Records for which the salary value is possibly not applicable, possibly satisfy the selection condition —to the extent reflected by the EPTV— (records 'P06' and 'P10'). All other records certainly do not satisfy the selection condition.

Query 5

SELECT *PID*, *Salary* FROM *Poll* WHERE *Salary* IS UNK

With query 5 the database is queried for records in which the salary value is *compatible* with the label 'UNK'. This means that a regular attribute value for salary must exist and there are no further restrictions imposed on this value by the query, i.e., the query searches for participants with a regular salary value, whatever this value is. Thus, it is important to notify that the query does not only search for participants in the database that have an unknown, but existent salary. Assume that the query is executed on the example database presented in table 5.1. The result of the

PID	*Salary*	*EPTV*
P01	2.600	$\{(T,1)\}$
P02	4.000±1000	$\{(T,1)\}$
P03	2.800	$\{(T,1)\}$
P04	N/A	$\{(\perp,1)\}$
P05	3.000±500	$\{(T,1)\}$
P06	'not_average'	$\{(T,1),(\perp,1)\}$
P07	UNK	$\{(T,1)\}$
P08	UNK	$\{(T,1)\}$
P09	N/A	$\{(\perp,1)\}$
P10	UNA	$\{(T,1),(\perp,1)\}$

Table 5.7 Result of Query 5.

query is then given in table 5.7. Records for which the salary value completely differs from 'N/A' completely certainly satisfy the selection condition (records 'P01', 'P02', 'P03', 'P05', 'P07' and 'P08'). For records with a salary value 'N/A' the corresponding EPTV represents 'not applicable' (records 'P04' and 'P09'). For records with a salary value that is possibly not applicable, the resulting EPTV is 'possibly true, possibly not applicable' (records 'P06' and 'P10').

Query 6

SELECT *PID*, *Salary* FROM *Poll* WHERE *Salary* IS *UNA*

With query 6 the database is finally queried for records in which the salary value is *compatible* with the label '*UNA*'. Such a selection condition imposes no restrictions on the database, which means that all records completely satisfy the selection condition. This is due to the fact that all domain values are completely certainly compatible with the uniform normalized possibility distribution that is denoted by the label '*UNA*' which represents 'no information available' (or all values are equally possible).

5.8 Frameworks for fuzzy querying

For the practical application of the querying techniques introduced in this chapter, one must have an implementation of a 'fuzzy' database model as for example the ones that are presented in chapter 4. Such an implementation is also called a 'fuzzy' database management system. A requirement of a 'fuzzy' database management system is that it must be fine tuned and integrated with the underlying logical framework that is chosen and used to express query satisfaction. As an illustration, we further describe 'fuzzy' querying of 'fuzzy' relational databases in subsection 5.8.1 and 'fuzzy' querying of 'fuzzy' object oriented databases in subsection 5.8.2. For both the relational and an object oriented database modelling, we present as well the possibilistic approach as the extended possibilistic approach.

5.8.1 'Fuzzy' relational databases

5.8.1.1 Possibilistic approach

An example of a 'fuzzy' relational database model with a possibilistic approach is the possibilistic relational model as originally presented in [16], that we already described in subsection 4.2.1.

Structural aspects.

As already has been described in subsection 4.2.1, in the possibilistic relational model all database and result relations r are extended with two extra attributes with names 'Pos' (possibility) and 'Nec' (necessity) of which the values express membership grades? In this way, two fuzzy sets are defined over the all tuples t of r: a fuzzy set with membership grades $\mathrm{Pos}(r)(t)$ which defines the tuples that possibly satisfy the predicate of relation r, and a fuzzy set with membership grades $\mathrm{Nec}(r)(t)$ which defines the tuples that necessarily satisfy the predicate of relation r. This extension is required to be able to guarantee the *closeness property* of the generalized relational algebra (see 'operational aspects' below).

Additionally, the users can be provided the option to specify threshold values

$$0 < \tau_{\mathrm{Pos}} \leq 1 \text{ and } 0 < \tau_{\mathrm{Nec}} \leq 1$$

with the query specification. If this is the case, then only tuples for which both $\mathrm{Pos}(r)(t) \geq \tau_{\mathrm{Pos}}$ and $\mathrm{Nec}(r)(t) \geq \tau_{\mathrm{Nec}}$ hold, are preserved in the query result.

Operational aspects.

The operators

- *union*
- *difference*
- *Cartesian product*
- *selection*
- *projection*

of the minimal subset of operators of the relational algebra, as originally presented by E.F. Codd [7] could be generalized as follows:

- *Union.*

$$e(\text{union}(r,r'))(t) = (\text{Pos}(\text{union}(r,r'))(t), \text{Nec}(\text{union}(r,r'))(t))$$

with

- $\text{Pos}(\text{union}(r,r'))(t) = \max(\text{Pos}(r)(t), \text{Pos}(r')(t))$
- $\text{Nec}(\text{union}(r,r'))(t) = \max(\text{Nec}(r)(t), \text{Nec}(r')(t))$

where r and r' represent relations, t is a tuple of r or r', and $\text{Pos}(r)(t)$ and $\text{Nec}(r)(t)$ respectively denote the degree of possibility and degree of necessity that t belongs to r.

- *Difference.*

$$e(\text{difference}(r,r'))(t) = (\text{Pos}(\text{difference}(r,r'))(t), \text{Nec}(\text{difference}(r,r'))(t))$$

with

- $\text{Pos}(\text{difference}(r,r'))(t) = \min(\text{Pos}(r)(t), 1 - \text{Pos}(r')(t))$
- $\text{Nec}(\text{difference}(r,r'))(t) = \min(\text{Nec}(r)(t), 1 - \text{Nec}(r')(t))$

where r and r' represent relations, t is a tuple of r or r', and $\text{Pos}(r)(t)$ and $\text{Nec}(r)(t)$ respectively denote the degree of possibility and degree of necessity that t belongs to r.

- *Cartesian product.*

$$e(\text{Cart-prod}(r,r'))(tt') = (\text{Pos}(\text{Cart-prod}(r,t'))(tt'), \text{Nec}(\text{Cart-prod}(r,r'))(tt'))$$

with

- $\text{Pos}(\text{Cart-prod}(r,r'))(tt') = \min(\text{Pos}(r)(t), \text{Pos}(r')(t'))$
- $\text{Nec}(\text{Cart-prod}(r,r'))(tt') = \min(\text{Nec}(r)(t), \text{Nec}(r')(t'))$

where r and r' represent relations, t is tuple of r, t' is a tuple of r' and $\text{Pos}(r)(t)$ and $\text{Nec}(r)(t)$ respectively denote the degree of possibility and degree of necessity that t belongs to r.

- *Selection.*

$$e(\text{selection}(r,c))(t) = (\text{Pos}(\text{selection}(r,c))(t), \text{Nec}(\text{selection}(r,c))(t))$$

with

- $\text{Pos}(\text{selection}(r,c))(t) = \min(\text{Pos}(r)(t), \text{Pos}(c)(t))$
- $\text{Nec}(\text{selection}(r,c))(t) = \min(\text{Nec}(r)(t), \text{Nec}(c)(t))$

where r represents a relation, t is a tuple of this relation, c is a (fuzzy) selection condition and $\text{Pos}(c)(t)$ and $\text{Nec}(c)(t)$ respectively denote the degree of possibility and degree of necessity that t satisfies c. These degrees are computed as described in section 5.3. Furthermore, $\text{Pos}(r)(t)$ and $\text{Nec}(r)(t)$ remain the degree of possibility and degree of necessity that t belongs to r.

- *Projection.*

$$e(\text{projection}(r,V))(v) = (\text{Pos}(\text{projection}(r,V))(v), \text{Nec}(\text{projection}(r,V))(v))$$

with

- $\text{Pos}(\text{projection}(r,V))(v) = \max_r \text{Pos}(r)(vw)$
- $\text{Nec}(\text{projection}(r,V))(v) = \max_r \text{Nec}(r)(vw)$

where r represents a relation, V is a subset of set X of all attributes of r, v takes values from V and w takes values from $X \setminus V$. The values $\text{Pos}(r)(vw)$ and $\text{Nec}(r)(vw)$ respectively represent the degree of possibility and degree of necessity that $t = vw$ belongs to r.

Remark that the computation rules given above are only an example of a possible way to generalize the operators of the relational algebra. As such, for the operators union, difference and projection a strict equality operation for attribute values is assumed, which implies that for the equality of fuzzy sets —representing possibility distributions— the standard equality operator (definition 2.11) is used to check whether two tuples are redundant, so that one of them has to be discarded from the result of the operation. An alternative, less stringent approach is possible here. Two possibility distributions π and π', that are both defined on the domain of an attribute A, can hereby approximately be considered as being equal if it holds that

$$\sup_{x \in dom_A} |\pi(x) - \pi'(x)| \leq \varepsilon_{dom_A}$$

where ε_{dom_A} is a given threshold value, specified to act on the values of attribute A.

5.8.1.2 Extended possibilistic approach

The structural and operational extensions for the approach with extended possibilistic truth values that have been described in subsection 3.4.1.2 can be applied with ‘fuzzy’ querying of ‘fuzzy’ databases, on condition that the generalized evaluation functions for ‘fuzzy’ querying conditions, described in section 5.5, are taken into account.

Example 5.6. Considering the relational counterpart of the ‘fuzzy’ artworks database presented in figure 5.1, the same ‘fuzzy’ query as in example 3.7 can be formulated, i.e.:

Give the name of the painting and the name of the artist of all very expensive paintings of artists who die at the beginning of the twentieth century, where the condition on the year of death of the artist must have significant larger impact on the query result than the condition on the value of the painting.

This query results in the following algebraic expression:

$$\begin{aligned}
&project(select(Cart-prod(Painting,Artist),\\
&\quad(Painting.Artist = Artist.Name, weight = 1)\ AND\\
&\quad(Painting.Value\ IS\ \text{‘}very\_expensive', weight = 0.6)\ AND\\
&\quad(Artist.Year\_of\_death\ IS\ \text{‘}beginning\_of\_twentieth\_century', weight = 1)),\\
&\quad\{Painting.Name, Artist.Name\})
\end{aligned}$$

where ‘very_expensive’ and ‘beginning_of_twentieth_century’ are linguistic terms that are modelled by the fuzzy sets with membership functions

$$\mu_{very\_expensive}(x) = \begin{cases} 0 & \text{iff } x < 10M \\ \dfrac{x-10}{10} & \text{iff } 10M \leq x \leq 20M \\ 1 & \text{iff } x > 20M \end{cases}$$

and

$$\mu_{beginning\_of\_twentieth\_century}(x) = \begin{cases} 1 & \text{iff } 1900 \leq x \leq 1910 \\ 0 & \text{iff } x < 1900 \text{ or } x > 1940 \\ \dfrac{1940-x}{30} & \text{iff } 1910 \leq x \leq 1940 \end{cases}$$

The Cartesian product results in a relation with 16 tuples. The first condition that must be evaluated is the generalized join condition

$$Painting.Artist = Artist.Name$$

This join condition is a simple condition of the form $A\ \theta\ B$ with θ being the equality operator. Only 5 tuples satisfy this condition. These are presented as an intermediate result in table 5.8 (for the sake of simplicity only the attributes representing the artist of the painting ($Painting.Artist$) and the name of the artist ($Artist.Name$) — which are involved in the join condition— and the resulting EPTV ($EPTV_1$) are presented. Because of the fact that the join condition has an associated weight 1 and because of the fact that there are only conjunction operators in the composite selection condition, the query processing can be continued departing with these 5 tuples.

Table 5.8 First intermediate results of the query processing.

$Painting.Artist$	$Artist.Name$	$EPTV_1$
$\{(Monet, 1)\}$	Monet	$\{(T, 1)\}$
$\{(Degas, 1)\}$	Degas	$\{(T, 1)\}$
$\{(DaVinci, 1)\}$	Da Vinci	$\{(T, 1)\}$
$\{(Ensor, 1), (Permeke, 0.4)\}$	Ensor	$\{(T, 1)\}$
$\{(Ensor, 1), (Permeke, 0.4)\}$	Permeke	$\{(T, 0.4), (F, 0.6)\}$

The evaluation of the other simple ‘fuzzy’ conditions

$$Painting.Value\, IS\, ‘very\_expensive'$$

and

$$Artist.Year\_of\_death\, IS\, ‘beginning\_of\_twentieth\_century'$$

respectively results in the EPTVs $EPTV_2$ and $EPTV_3$ as presented in table 5.9. For the computations of these EPTVs, the following definitions for the linguistic terms have been used:

- Linguistic term ‘about_15M’:

$$\pi_{about\_15M}(x) = \begin{cases} \frac{x-14.8}{0.2} & \text{iff } 14.8M \leq x \leq 15M \\ \frac{15.2-x}{0.2} & \text{iff } 15M < x \leq 15.2M \\ 0 & \text{else} \end{cases}$$

- Linguistic term ‘more_than_8M’:

$$\pi_{more\_than\_8M}(x) = \begin{cases} 1 & \text{iff } x \geq 8M \\ 0 & \text{else} \end{cases}$$

- Linguistic term ‘very_expensive’:

$$\pi_{very\_expensive}(x) = \begin{cases} 0 & \text{iff } x < 10M \\ \frac{x-10}{10} & \text{iff } 10M \leq x \leq 20M \\ 1 & \text{iff } x > 20M \end{cases}$$

- Linguistic term ‘at_least_1K’:

$$\pi_{at\_least\_1K}(x) = \begin{cases} 1 & \text{iff } x \geq 1K \\ 0 & \text{else} \end{cases}$$

- Linguistic term ‘around_1519’:

$$\pi_{around\_1519}(x) = \begin{cases} \frac{x-1517}{2} & \text{iff } 1517 \leq x \leq 1519 \\ \frac{1521-x}{2} & \text{iff } 1519 < x \leq 1521 \\ 0 & \text{else} \end{cases}$$

Using the extended conjunction operator $\wedge^{w}{}_{prob}$ that is described in example 3.6 while considering the given weights $w_1 = 1$, $w_2 = 0.6$ and $w_3 = 1$ allows us to find the aggregated intermediate results that are presented in table 5.10. For each of the obtained tuples, the associated aggregated weight is $w = 1$.

Table 5.9 Second intermediate results of the query processing.

$Artist.Name$	$Value$	$EPTV_2$	$Year\_of\_death$	$EPTV_3$
Monet	about_15M	$\{(T,0.51),(F,0.51)\}$	1926	$\{(T,0.47),(F,0.53)\}$
Degas	more_than_8M	$\{(T,1),(F,1)\}$	1917	$\{(T,0.77),(F,0.23)\}$
Da Vinci	very_expensive	$\{(T,1),(F,0.5)\}$	around_1519	$\{(F,1)\}$
Ensor	at_least_1K	$\{(T,1),(F,1)\}$	1949	$\{(F,1)\}$
Permeke	at_least_1K	$\{(T,1),(F,1)\}$	1952	$\{(F,1)\}$

Table 5.10 Aggregated intermediate results.

$Artist.Name$	$EPTV_1 \wedge^w{}_{prob} EPTV_2 \wedge^w{}_{prob} EPTV_3$
Monet	$\{(T,0.24),(F,0.77)\}$
Degas	$\{(T,0.77),(F,1)\}$
Da Vinci	$\{(F,1)\}$
Ensor	$\{(F,1)\}$
Permerke	$\{(F,1)\}$

The final result, which is obtained after applying the projection operator and normalization of the resulting EPTV, is given in table 5.11.

Table 5.11 Final query result.

$Painting.Name$	$Artist.Name$	μ_T	μ_F	$\mu_\perp$
'Fishermans house'	'Monet'	0.31	1	0
'The ballet course'	'Degas'	0.77	1	0

◇

5.8.2 *'Fuzzy' object oriented databases*

Also the object oriented and object relational database models have been extended and generalized to 'fuzzy' database models. Hereby, as well the possibilistic as the extended possibilistic approach have been used as the underlayin framework. All considerations with respect to both the structural and behavioural aspects given in subsection 3.4.2 remain valid.

An example of a 'fuzzy' object oriented database model that is based on the possibilistic approach is the FOOD-model [1, 14, 2, 3] that has been described in subsection 4.2.3. The 'fuzzy' constraint based object oriented database model [10], presented in subsection 4.2.4, is an example of a database model that is supported by the extended possibilistic approach.

The FOOD-model has been implemented [4].

References

1. G. Bordogna, D. Lucarella, and G. Pasi, “A Fuzzy Object Oriented Data Model”, in: *Proceedings of the Third IEEE International Conference on Fuzzy Systems, FUZZ-IEEE’94*, Orlando, USA (1994) 313–318.
2. G. Bordogna, D. Lucarella, and G. Pasi, “Extending a Graph-Based Data Model to Manage Fuzzy and Uncertain Information”, in: R. De Caluwe (ed.), *Fuzzy and Uncertain Object-Oriented Databases: Concepts and Models*, World Scientific, Signapore (1997) 97–122.
3. G. Bordogna, G. Pasi, and D. Lucarella, “A Fuzzy Object-Oriented Data Model for Managing Vague and Uncertain Information”, *International Journal of Intelligent Systems* **14** 7 (1999) 623–651.
4. G. Bordogna, A. Leporati, D. Lucarella, and G. Pasi, “The Fuzzy Object-Oriented Database Management System”, in: G. Bordogna, and G. Pasi (eds.), *Recent Issues on Fuzzy Databases*, Physica-Verlag, Heidelberg, Germany (2000) 209–236.
5. P. Bosc, D. Kraft, and F. Petry, “Fuzzy sets in database and information systems: status and opportunities”, *Fuzzy Sets and Systems* **153** 3 (2005) 418–426.
6. B. Buckles, F. Petry, and H. Sachar, “A Domain Calculus for Fuzzy Relational Databases”, *Fuzzy Sets and Systems* **29** (1989) 327–340.
7. E.F. Codd, “Relational Completeness of Data Base Sublanguages”, in: *Data Base Systems, Courant Computer Science Symposia Series 6*, R.J. Rustin (ed.) Prentice-Hall, Englewood Cliffs, USA (1972).
8. G. De Tré, “Extended Possibilistic Truth Values”, *International Journal of Intelligent Systems* **17** (2002) 427–446.
9. G. De Tré, R. De Caluwe, J. Kacprzyk, and S. Zadrozny, “On flexible querying via extensions to fuzzy sets”, in: *Proceedings of the 4th Conference of the European Society for Fuzzy Logic and Technology (Eusflat 2005) and 11th Rencontres Phrancophones sur la logique floue et ses applications (LFA’05)*, Barcelona, Spain, (2005) 1225–1230.
10. G. De Tré, and R. De Caluwe, “A Constraint Based Fuzzy Object Oriented Database Model”, in: Z. Ma (ed.), *Advances in Fuzzy Object-Oriented Databases: Modeling and Applications*, Idea Group Publishing, Hershey, USA, 2005, 1–45.
11. D. Dubois, and H. Prade, *Possibility Theory* (Plenum Press, New York, USA, 1988).
12. D. Dubois, H. Fargier, and H. Prade, “Beyond min aggregation in multicriteria decision: (ordered) weighted min, discri-min and leximin”, in: *The ordered weighted averaging operators: Theory and applications.*, R.R. Yager, and J. Kacprzyk (eds.) (Kluwer Academic Publishers, Boston, USA, 1997) 181–192.
13. D. Dubois, and H. Prade, “Possibility theory, probability theory and multiple-valued logics: A clarification”, *Annals of Mathematics and Artificial Intelligence* **32** (2001) 35–66.

14. D. Lucarella, G. Bordogna, and G. Pasi, “Pattern-Based Retrieval in a Fuzzy Object-Oriented Data Base”, in: *Proc. of the Tenth Annual ACM Symposium on Applied Computing, ACM-SAC’95*, Nashville, USA (1995) 508–513.
15. F.E. Petry, *Fuzzy Databases: Principles and Applications* (Kluwer Academic Publishers, Boston, USA, 1996).
16. H. Prade, and C. Testemale, “Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries”, *Information Sciences* **34** (1984) 115–143.
17. L.A. Zadeh, “Fuzzy sets as a basis for a theory of possibility”, *Fuzzy Sets and Systems* **1** 1 (1978) 3–28.