KUNGL TEKNISKA HÖGSKOLAN

Indiscernibility and Vagueness in Spatial Information Systems

Karim Oukbir

Stockholm 2003

Doctoral Dissertation

Royal Institute of Technology

Department of Numerical Analysis and Computer Science

Akademisk avhandling som med tillstånd av Kungl Tekniska Högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen 10 november 2003 kl 13.00 i Kollegiesalen, Administrationsbyggnaden, Kungl Tekniska Högskolan, Valhallavägen 79, Stockholm.

ISBN 91-7283-602-4
TRITA-NA-0321
ISSN 0348-2952
ISRN KTH/NA/R-03/21-SE

Universitetsservice US AB, Stockholm 2003

Abstract

We investigate the use of the concept of indiscernibility and vagueness in spatial information systems. To represent indiscernibility and vagueness we use rough sets, respectively fuzzy sets.

We introduce a theoretical model to support approximate queries in information systems and we show how those queries can be used to perform uncertain classifications. We also explore how to assess quality of uncertain classifications and ways to compare those classifications to each other in order to assess accuracies.

We implement the query language in an SQL relational language to demonstrate the feasibility of approximate queries and we perform an experiment on real data using uncertain classifications.

ISBN 91-7283-602-4 • TRITA-NA-0321 • ISSN 0348-2952 • ISRN KTH/NA/R-03/21-SE

Sammanfattning

Vi undersöker användningen av begreppen oskiljbarhet och vaghet i spatiella informationssystem. Vi använder oss av grova mängder (rough sets) för att modellera oskiljbarhet och oskarpa mängder (fuzzy sets) för att modellera vaghet.

Vi definierar en teoretisk modell för ett frågespråk som hanterar osäker information, och med hjälp av denna modell introducerar vi ett nytt sätt att utföra osäkra klassificeringar. Vi undersöker också nya sätt att mäta och jämföra noggrannheten hos osäkra data.

Vi implementerar vår modell i ett SQL-baserat frågespråk för att undersöka möjligheten att utföra approximativa frågor i informationssystemet samt genomför slutligen ett experiment med verkliga data.

ISBN 91-7283-602-4 • TRITA-NA-0321 • ISSN 0348-2952 • ISRN KTH/NA/R-03/21-SE

Résumé

Nous explorons l'utilisation de la notion d'indiscernabilité et celle de la notion du flou dans les systèmes d'information spatiales. Nous utilison la théorie des ensembles approximatifs pour modeler l'indiscernabilité des éléments de l'univers et la théorie des ensembles flous pour la representation de l'information floue.

Nous introduisons un modèle théorique qui permet d'implémenter les interrogations approximatifs dans les systèmes d'information. Ce nouveau langage permet de définir le concept de classification incertaines. De plus, nous explorons les moyen de calculer et de comparer les degrés d'exactitudes de ces classifications.

Nous implementons ce langage d'interrogation à l'aide d'un langage de requête SQL pour démontrer la faisabilite de notre idée et nous effectuons des expériences sur des données réels.

ISBN 91-7283-602-4 • TRITA-NA-0321 • ISSN 0348-2952 • ISRN KTH/NA/R-03/21-SE

Acknowledgments

First of all I would like to thank my supervisor Per Svensson for his unlimited help and his support, for insightful discussions on the ideas of this thesis, and for quick feedback on my manuscripts. Without Per this thesis would not have been.

I also thank Stefan Arnborg, my secondary supervisor, for his help and his support.

Ola Ahlqvist and Johannes Keukelaar for our joint papers, collaborating with them has always been a pleasure. It was also a great help to be able to discuss with Johannes some of the ideas of this manuscript.

Mike Worboys was the opponent for my licentiate degree in June 2001. The discussions I had the opportunity to have with him, both here in Stockholm and in Norway during the ScanGIS'2001 conference, have been a great help when preparing my doctoral thesis. Thank you Mike!

Johannes Keukelaar (again), Jesper Fredriksson, Klas Wallenius and Öjvind Johansson for great time past here.

I am grateful to Lars Engebretsen who has helped me with LaTeX matters when writing this manuscript. All the other members of my research group, those that are still in the group and those that have left it, they all have been very helpful to me.

I wish to thank Yiyu Yao who, through email, has helped me understanding some issues related to interval sets and rough sets. Tore Risch and Timour Katchaounov for many fruitful discussions and their help with the AMOS II system. Ralf Hartmut Güting and his group for interesting discussions in the beginning of my thesis work.

I also thank the former Center for Geoinformatics, Stockholm and the Swedish Defense Research Agency (FOI) for providing fundings for this research. The Taylor & Francis Group (www.tandf.co.uk) for kindly allowing me to reuse the materials presented in 7.3 from our paper [AKO03].

My parents, my sister and my cousin Omar, thank you for your support.

My lo♡ely fiancé Jennie, thank you for your infinite patience!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An Information System is a piece of software that stores information about some objects and allows retrieving and managing that information. A university information system may store information about students, courses and teachers, the courses that each student is registered to and the courses that each teacher teaches for instance. Besides allowing storage and retrieval of information, an information system also provides a means to combine existing information to derive new information. For instance, to derive the total number of students registered to a given course or to list the courses with the highest number of students.

Spatial information systems deal with information about objects located in a spatial frame such as the Earth's surface. The spatial objects may be represented in the system, e. g. , as polygons or groups of pixels. They may represent areas on the Earth together with some information of interest. An example would be polygons representing counties, points representing cities together with some demographic information for each city.

Assume we wish to retrieve the city with the largest population. This task is not spatial in nature, however, imagine we are interested in listing the city with the largest population that is located within 100 kilometers from the capital. This latter task is spatial in nature as locations of cities are involved. A spatial information system should provide a means to perform such tasks, i. e. to store, retrieve and combine spatial information.

Note that in a spatial information system the spatial frame of reference may as well be a silicon chip or the human body. Basically, anything provided it has a spatial extent.

Early traces of use of (manual) spatial information systems go back to the 17th century. According to history, as pointed out in [SC03], in the year 1855, when the Asiatic cholera was decimating the population of London, an epidemiologist marked all locations on a map where the disease struck. He later discovered that the locations formed a cluster whose centroid turned out to be a water pump. Once the water pump was shut down, the disease began to disappear.

Today, information systems are widely used in many different sectors. Data, i. e. raw information, is gathered at an incredible rate. NASA's Earth Observing System gathers one terabyte of data every day. We gather data and we produce information. The Digital Earth project (www.digitalearth.gov), for instance, is an attempt to build a virtual representation of our planet using natural and cultural information about the Earth.

Although the area of spatial information have received much attention lately, there are still areas that need to be investigated. Uncertainty handling in spatial information systems is one such example [Goo93].

Uncertainty is an important aspect of both thematic and spatial information. Any information about some natural feature stored in computer is inherently uncertain as computer representations can only partially agree with reality.

On one hand, a computer representation of some natural phenomenon is only a snapshot of reality. For instance, the boundary of a lake may change over time, vegetation may disappear. Also, our perception of reality together with our knowledge may influence the information. For instance, the discovery of some new vegetation type may render a computerized map imprecise if its vegetation parts become outdated.

On the other hand, from the hardware side, we are facing sampling limitations, limited measurement precision, instrument error et cetera ...

It is a matter-of-fact that we have to deal with uncertainty.

Demographic information stored in a computer is a typical example of uncertain thematic information. It is inherently imprecise because it can never be recent enough. Now, assume we make the computer achieve the following task: retrieve the city with the largest population within 100 kilometers from the capital. The answer that is returned is in accordance with the information that is stored but might not agree with reality. The impact of such a result on the users of an information system depends on the degree of confidence they have in the result.

One solution to this problem is to model uncertain information in the system.

If we succeed in modeling the uncertainty of the information in the system, in this case taking into account the fact that the population size may have changed since the latest census, then the system may allow an uncertain output. The output may be: 'city A **OR** city B has the largest population within 100 kilometers from the capital'.

This is the topic of my research.

There are two different models of uncertainty that I have chosen to work with: vagueness and indiscernibility. These two types of uncertainties are commonly used in everyday life.

It is common to everyone to reason with *vague* information, for instance when we are told that a person is *young*. Young is clearly a vague concept as it is not possible to define a sharp boundary for how old a young person is. Yet we reason extensively with vague information: "he is too young to drink", "the bride is younger than the groom" et cetera ...

It is also common to everyone to reason with *indiscernible* concepts. For instance *color* is an example of such a concept. Imagine trying to explain how lime green color differs from medium sea green for someone who only knows of the green color. How do we describe the difference between lime green and medium sea green colors if there are no words for them? According to [Lin01] there are languages which allow to discern black and white colors only. Yet we use colors extensively in our reasoning: “Lime green is refreshing”, “red is warm” to name a few.

Combining vague and indiscernible information is also common usage. For instance “he bought an old lime green car” and “he bought an older medium sea green car” may both be vague and indiscernible from each other.

One question remains, how do we make computers ‘understand’ vagueness and indiscernibility? To do so, some theoretical models have been developed which I introduce in Chapter 2 and study more extensively in Chapter 3.

Another issue is to interact with an information system that supports both vagueness and indiscernibility. That is, we need some kind of language to describe the tasks to be achieved by the information system. Johannes Keukelaar in his thesis [Keu02] proposes a visual language as an interaction model. In my thesis I propose a relational language. In a relational language a user defines the task to the system by describing the result that is desired. This is different from imperative languages where tasks are described by means of orders to the computer, also called instructions.

The relational language and the relational model have been introduced by Codd in [Cod70] in an attempt to facilitate computer interaction for business analysts. Later, researchers in the area of spatial information with the same motivation have presented ideas of using the same model for spatial information. See [Ege94, HSH92, HS93] for example. I had the same concern in my research, that is to facilitate interaction, which motivates the choice of the relational model.

First, I developed a theoretical model for a query language that can be used to interact with uncertain information, see Chapter 4. Then, in Chapter 6, I show how to implement this model in the relational model.

Due to the underlying theory that I have used to model uncertainty, there is an issue that has to be taken into account when interacting with uncertain information. The issue can be explained by the notion of truth-functionality, see Section 3.2. Intuitively, truth-functionality is the ability to compute a result from some input data, i. e. to combine information to obtain new information. This notion can be established as follows. Assume I say that “Stockholm is in Sweden **AND** Paris is in France”, then one could deduce that I have said nothing but the *truth*. Indeed, both sentences are true and they are combined with the AND operation to form another true statement. But now imagine that I say “I caught a cold **BECAUSE** I took off my sweater”. Although both sentences might be correct, one cannot state if the combination actually is, even if it *seems* to be correct. Perhaps it is better grasped with a slight change of the example: “I took off my sweater BECAUSE I caught a cold”. Obviously, to state the truthfulness of this combined sentence it is not sufficient to look at the truthfulness of the sub-sentences. Hence, BECAUSE is not a truth-functional operation. Intuitively, the operations that I use in my query language behave in the same way as the BECAUSE operation, i. e. they do not allow combining information in a simple way. Under some circumstances, however, this issue can be avoided. In Section 3.2.3 I introduce a method to accomplish this.

An interesting question is what can be achieved with this new language. As I suggested in Chapter 4, a relation between thematic and spatial uncertainty may be built in this new model. The idea is that if I cannot describe an object with certainty (thematic or attribute uncertainty) then I cannot locate the object with certainty (spatial uncertainty). Imagine you are asked to look for a medium sea green colored car that is parked in the neighborhood. Unless you know how to recognize that shade of green, or that there is no other green car in the area in question, you would not be able to locate the car. You would be uncertain where the car is located since it could be any of the green cars each of which having a specific location and possibly a different shade of green.

In the same chapter, I also show that classifications can be expressed in a new way. Classification is a common task in real life. Pawlak [Paw91] claims that knowledge is deep-seated in the classificatory abilities of human beings and other species. But the ability to classify is closely related to our ability to describe things that surround us. We classify according to descriptions: for instance, often the blue color symbolizes (classifies to) something that is cold and the red color to something that is hot. In this thesis I show how to perform classification based on uncertain descriptions, see Chapter 4.

When manipulating uncertain information it is useful to have some kind of idea of how uncertain the information is. In Chapter 5 I present some quantitative measures to assess accuracies of uncertain information. There, I also provide some qualitative methods that are based on topological relations.

Finally, some real-life experiments are performed in Chapter 7.

1.1 Contributions

This thesis is partly based on three papers that have been written in collaboration with O. Ahlqvist and J. Keukelaar. Our first paper [AKO98], "Using Rough Classification to Represent Uncertainty in Spatial Data", introduces classifications using rough set techniques. The second paper [AKO00], "Rough classification and accuracy Assessment", is an extension of the first paper in which some more accuracy measures of rough classifications are developed. The third paper [AKO03], "Rough and fuzzy geographic data integration", discusses rough fuzzy classifications that are based on rough fuzzy sets.

The paper "AMORose, a realm based spatial database system" [Ouk97] presents some implementation ideas for the AMORose spatial database system.

The idea of extended class representatives of Section 3.2.3 is my own work.

Section 5.4 is based on a technical report [OKA01].

Chapter 4 and Chapter 6 are my own work.

Chapter 2

Uncertainty Representation

We overview some methods that may be used to represent indiscernibility and vagueness in spatial information systems.

Section 2.1 presents an ontology of imperfection, Section 2.2 and Section 2.3 introduce fuzzy sets and rough sets respectively. Section 2.4 introduces an alternative definition of rough sets that are called Marek-style rough sets, then in Section 2.5 we present an extension of Marek-style rough sets to support fuzziness.

2.1 An Ontology of Imperfection

As Worboys suggested in [WC01], natural phenomena that are mapped into the computer are subject to imperfections. Imperfections may be due to the *imprecision* of the instruments used to measure certain natural phenomenons, but also to our lack of ability to map certain concepts precisely.

Figure 2.1 shows the classification of imperfection suggested by Worboys. There are two classes of imperfections:

Inaccuracy, which is a description, or a mapping, of some natural phenomenon that does not agree with reality. There are two kinds of such inaccuracies:

- **Error of Commission**(EC) occurs when an element in the mapping does not exist in the real situation. For instance, a map of Sweden that contains the city of Paris.
- **Error of Omission** (EO) occurs when the mapping fails to report an element that does exist in the real situation. For instance, a map of Sweden that does not contain the city of Stockholm.

Imprecision refers to mappings of some natural phenomenons that are not represented exactly as they appear. In this thesis, we focus on the following two kind of imprecision:

Vagueness, which is an imprecision that characterizes concepts that cannot be sharply outlined. A person is *tall* is a typical such concept, for instance.

Indiscernibility, which is an imprecision that is due to our inability to distinguish some elements that build up a given natural phenomenon. For instance, red and green are indistinguishable for a color-blind.

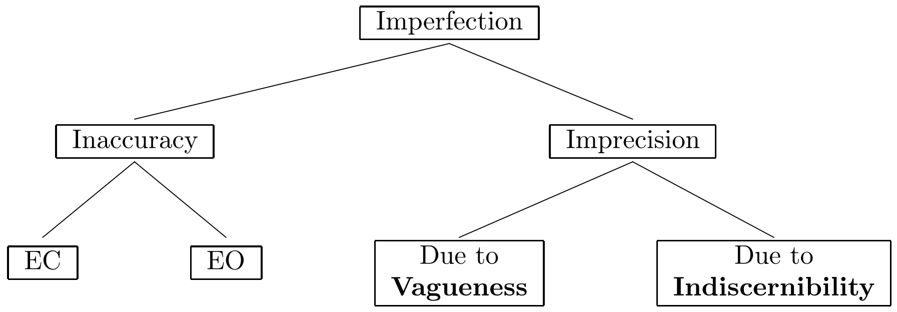


Figure 2.1. Ontology of Imperfection

2.2 Fuzzy Sets

Fuzzy sets have been introduced by L. A. Zadeh in [Zad65]. Fuzzy sets will be more extensively studied in Chapter 3.

Sometimes we wish to represent a concept that does not fit as a standard set. A vague concept, for instance, the concept *young*, may be represented as a set P of persons p. Young persons are members of the set P whereas all other persons, those that are not considered young, are not. However, a person does not become old from one day to another, rather, it is *gradual* process. It can be represented, as Zadeh suggested, using a membership function $\mu_P(p)$ which takes values between 0 and 1. The value 0 indicates that the person p is definitely not young. The value 1 for p indicates that the person in question is definitely young. Any value between 0 and 1 indicates a degree to which a person p is young. See Figure 2.2 for an example membership function.

In the example, a person's degree of membership of the set P decreases from the age of 20 and reaches 0 at the age of 30. The interpretation is that persons who are more than 30 years old are definitely not young, persons younger than 20 are definitely young. To any age between 20 and 30 corresponds a degree of youngness determined by the membership function.

Fuzzy sets are able to model inherent vagueness.

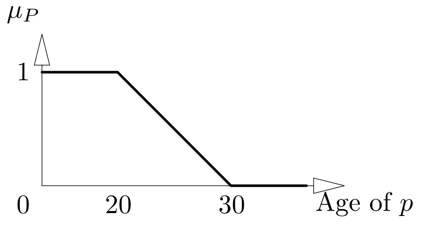


Figure 2.2. Example Fuzzy Membership Function

2.3 Rough Sets

Rough sets will be more extensively studied in Chapter 3.

As Pawlak suggests in his seminal paper [Paw82], it is possible to consider a universe $\mathcal{U}$ together with an equivalence relation $\theta \subseteq \mathcal{U} \times \mathcal{U}$. The equivalence relation indicates which elements of the universe are indiscernible from each other. These elements form an equivalence class. One can then build up an *approximation* of sets of elements $X \subseteq \mathcal{U}$ by using lower and upper approximations $\underline{X}$ and $\overline{X}$. $\underline{X}$ approximates X from below, i. e. $\underline{X}$ contains only those elements which belong to equivalence classes that are subsets of X. $\overline{X}$ contains elements which belong to equivalence classes that are not disjoint from X.

There are two problems with Pawlak's definition. First, intersection and union are not truth-functional [Yao96], i.e. the equalities $\underline{X} \cup \underline{Y} = \underline{X \cup Y}$ and $\overline{X \cap Y} = \overline{X} \cap \overline{Y}$ do not hold. Only $\underline{X} \cup \underline{Y} \subseteq \underline{X \cup Y}$ and $\overline{X \cap Y} \subseteq \overline{X} \cap \overline{Y}$ hold. The truth-functionality issue is treated in Section 3.2. The second problem with Pawlak's definition is that we often wish to construct a rough set in cases where we have only incomplete knowledge of the set X that we are trying to approximate [MT99]. Under Pawlak's definition, however, the construction of the rough set is closely tied to X which is assumed to be well defined. In the following sections we will present some less restrictive alternatives to Pawlak-style rough sets.

Yao et al. present different ways to define rough sets in [YWL97]. Rough set extensions to functions and relations have also been studied, see [Paw96]. In [Paw91], Pawlak presents some applications of rough sets to reasoning about data. Rough sets have also been studied in relation to fuzzy sets in [Paw85] and for rough fuzzy integration purposes in [DP92, Yao97].

It is worth noting that rough sets and fuzzy sets represent different aspects of the uncertainty of the information under considerations. Hence, they should be considered as complementary methods rather than competitive ones [DP90].

2.4 Marek's Model

In [MT99], Marek et al. have introduced a less restrictive rough set model based on Iwinski's work [Iwi87].

As for Pawlak-style rough sets, Marek's model requires an approximation space $(\mathcal{U}, \theta)$ where θ is an equivalence relation imposing a granularity on a finite universe $\mathcal{U}$. A *definable set* in this universe is a union of equivalence classes, i. e. sets containing indiscernible elements. A rough set, in the sense of Marek, is a pair of definable sets (L, U) such that $L \subseteq U$. Such a construction is an approximation of any set A such that $L \subseteq A \subseteq U$. Given $X = (L, U)$, we define the lower approximation of X, denoted $\underset{\sim}{X}$, and upper approximation of X, denoted $\widetilde{X}$, as $\underset{\sim}{X} = L$ and $\widetilde{X} = U$ respectively.

Obviously then, for a certain element $x \in \mathcal{U}$, if $x \notin U$, then $x \notin X$, if $x \in L$, then $x \in X$ and, finally, if $x \in U \setminus L$ (the area of uncertainty), then we are not sure if x belongs to X or not.

The difference between the above and Pawlak's definition of a rough set is that Pawlak defines a rough set as an approximation to a specific set, say X. Also, this rough set is a pair $(\underline{X}, \overline{X})$, where $\underline{X}$ (resp. $\overline{X}$) is the 'maximal' (resp. 'minimal') definable set that obeys $\underline{X} \subseteq X$, respectively $X \subseteq \overline{X}$.

Rough sets as defined here are a modification of Pawlak-style rough sets. They have truth-functional intersection and union operators as shown below.

$$(L, U) \cap (L', U') = (L \cap L', U \cap U')$$

$$(L, U) \cup (L', U') = (L \cup L', U \cup U')$$

Where $X = (L, U)$ and $Y = (L', U')$.

If (L, U) approximates a given crisp set A, and (L', U') approximates a given crisp set B, then $(L \cap L', U \cap U')$ approximates $A \cap B$ and $(L \cup L', U \cup U')$ approximates $A \cup B$. The approximation operations may be defined as shown below.

$$\underset{\sim}{X \cap Y} = \underset{\sim}{X} \cap \underset{\sim}{Y} \text{ and } \underset{\sim}{X \cup Y} = \underset{\sim}{X} \cup \underset{\sim}{Y}$$

$$\widetilde{X \cap Y} = \widetilde{X} \cap \widetilde{Y} \text{ and } \widetilde{X \cup Y} = \widetilde{X} \cup \widetilde{Y}$$

2.5 Rough Fuzzy Extension of Marek's Model

In [AKO03], we have introduced a particular rough-fuzzy hybrid model based on Marek's work.

Dubois and Prade [DP92] have proposed the concept of rough fuzzy set, which is a Pawlak-style rough approximation of a fuzzy set. The following definition is based on their work, modified to correspond to the definition of Marek-style rough sets.

To integrate fuzzy sets and rough sets, we define a *rough fuzzy definable set*, or RF-*definable set*. An RF-definable set is a fuzzy set defined by the membership function $\mu : \mathcal{U} \rightarrow [0, 1]$ such that $\forall x \forall y \in [x] : \mu(y) = \mu(x)$. Here, $[x]$ represents the equivalence class of x, i. e. the set of elements that cannot be distinguished from x. A *rough fuzzy set* (RF-set) is a pair of RF-definable sets (L, U), such that $L \subseteq U$. We say that (L, U) approximates the fuzzy set F if $L \subseteq F \subseteq U$. Given an RF-set $X = (L, U)$, then we define $\underset{\sim}{X} = L$ and $\widetilde{X} = U$. To prevent confusion with rough sets, RF-sets will often be written, in a slight abuse of notation, as $\mu_X = (\mu_L, \mu_U)$, or, equivalently, $\mu_X = (\mu_{\underset{\sim}{X}}, \mu_{\widetilde{X}})$.

Intersection and union of two RF-sets μ_X and μ_Y are defined as:

$$\mu_X \cap \mu_Y = (\min(\mu_{\underset{\sim}{X}}, \mu_{\underset{\sim}{Y}}), \min(\mu_{\widetilde{X}}, \mu_{\widetilde{Y}}))$$

$$\mu_X \cup \mu_Y = (\max(\mu_{\underset{\sim}{X}}, \mu_{\underset{\sim}{Y}}), \max(\mu_{\widetilde{X}}, \mu_{\widetilde{Y}}))$$

2.6 The Egg-Yolk Model

The egg-yolk model has been presented in [LC95]. Initially, the egg-yolk model was developed to allow the representation of some concept, or class, into class members and typical class members. The set of typical members is the *yolk*, the larger set of all members of the class is the *egg*, see Figure 2.3.

For instance, one may define the class of all vehicles as follows: a vehicle is a device by which any person or property may be propelled, or moved, or drawn upon a highway, excepting a device moved exclusively by human power or used exclusively upon stationary rails or tracks. This definition excludes aircraft in the class of vehicles, i. e. aircraft are not part of the egg. The typical class members, i. e. the elements of the yolk, under this definition may be cars, trucks or buses for instance. For integration purposes, one may compare different class definitions to each other through their corresponding egg-yolks as shown in [LC95].

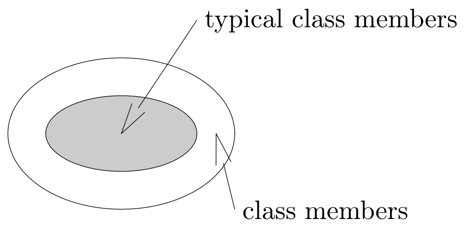


Figure 2.3. An example Egg-Yolk.

Some work has been done to apply the egg-yolk theory in a spatial context. For instance, Cohn et al. in [CG96] show how to use the model to represent regions with indeterminate boundaries. Ways to represent relations between regions with indeterminate boundaries using the egg-yolk model have been presented in [RS01].

Some studies comparing the rough set and the egg-yolk theories have been presented in [BP01].

2.7 Other Models

Interval set is an extension of standard set theory that may be used for uncertain reasoning. There are several equivalent definitions of interval sets [Iwi87, MT99, Yao96]. We pick the one presented by Yao. Let $\mathcal{U}$ be a finite set, called the universe, and $2^{\mathcal{U}}$ its power set. A subset of $2^{\mathcal{U}}$ of the form $[I, I'] = \{X \in 2^{\mathcal{U}} | I \subseteq X \subseteq I'\}$ is called an *interval set*. The set I is called the *lower bound* of the interval set and the set I' the *upper bound*.

In [SW98], Stell et al. have specified a model that is closely related to rough sets called vague regions. A vague region is also defined over a partition. Let R be a set of elements considered as basic regions. The basic regions may be cells in a subdivision of physical space or pixels. For a given partition A of R defined through the function $\sigma : R \rightarrow A$, one can define a vague region with the function $v : A \rightarrow \{\text{in}, \text{maybe}, \text{out}\}$. The set of basic regions *definitely* included in the vague regions is defined as:

$$\text{lower}(v) = \{r \in R | v\,\sigma\,r = \text{in}\}$$

The set of basic regions that is *possibly* included in the vague regions is defined as:

$$\text{upper}(v) = \{r \in R | v\,\sigma\,r \neq \text{out}\}$$

As stated in [SW98], a vague region can be thought of as standing for any region that lies between the union of the elements of lower(v) and the union of the elements of upper(v).

2.8 Conclusion

We have presented different methods that can be used to represent uncertainty when mapping natural phenomena into the computer.

In Section 2.5 we have presented our extension [AKO03] of Marek's model to support vagueness. It turns out that if we define a partial order on Marek-style rough sets presented in Section 2.4, then Marek-style rough sets can be compared to Pawlak-style rough sets. To do so, we define the knowledge ordering $\preceq_{kn}$ [MT99] on pairs of definable sets (L, U) as:

$$(L_1, U_1) \preceq_{kn} (L_2, U_2) \text{ if } L_1 \subseteq L_2 \text{ and } L_1 \subseteq L_2$$

In this case, if (L, U) approximates the set X, then, $(L, U) \preceq_{kn} (\underline{X}, \overline{X})$. The latter statement means that Pawlak-style rough sets are $\preceq_{kn}$-maximal Marek-style rough sets. As a consequence, Pawlak-style rough sets may be better approximations than Marek-style rough sets. We therefore focus on Pawlak-style rough sets and their integration with fuzzy sets in the remainder of this thesis.

Chapter 3

Indiscernibility and Vagueness

We study the theory of rough sets and the theory of fuzzy sets and their integration into a unified model called rough fuzzy sets.

Section 3.1 presents approximation of sets which is the theoretical basis of rough sets. Section 3.2 investigates the truth-functionality issue related to the utilization of rough sets. Section 3.4 shows how to approximate fuzzy sets.

3.1 Approximation of Sets

Rough sets were first introduced by Pawlak [Paw82] and are based on approximation spaces. An approximation space is a pair $\mathcal{A} = (\mathcal{U}, \theta)$. Here, θ is an equivalence relation, also called *indiscernibility* relation, imposing a granularity on the universe $\mathcal{U}$ such that $\theta \subseteq \mathcal{U} \times \mathcal{U}$. Furthermore, we assume $\mathcal{U}$ to be finite. For $x \in \mathcal{U}$, let $[x]_\theta$ be the equivalence class containing x, i. e. $[x]_\theta = \{y \mid y\,\theta\,x\}$. The element x is referred to as the *class representative* of the equivalence class $[x]_\theta$. The quotient set $\mathcal{U}/\theta$ is the set of all equivalence classes.

Given an arbitrary set $X \subseteq \mathcal{U}$, we wish to describe X in terms of elements or granules of $\mathcal{U}/\theta$. Pawlak proposed the use of *lower* and *upper approximations* of a set X, denoted $\underline{\mathcal{A}_\theta}(X)$ and $\overline{\mathcal{A}_\theta}(X)$, respectively. Lower and upper approximations are defined as:

$$\underline{\mathcal{A}_\theta}(X) = \{x \in \mathcal{U} \mid [x]_\theta \subseteq X\},$$

$$\overline{\mathcal{A}_\theta}(X) = \{x \in \mathcal{U} \mid [x]_\theta \cap X \neq \emptyset\}.$$

The semantics of the approximations of sets may be defined as follows:

- elements of the universe that belong to $\underline{\mathcal{A}_\theta}(X)$ are those elements that *surely belong* to the set X,
- elements that belong to $\overline{\mathcal{A}_\theta}(X)$ *possibly* belong to the set X,

- elements that belong to $\mathcal{U} \setminus \overline{\mathcal{A}_\theta}(X)$ are elements of the universe that *surely do not belong* to the set X.

Hence, the uncertainty lies in $\overline{\mathcal{A}_\theta}(X) \setminus \underline{\mathcal{A}_\theta}(X)$ which is also called *area of uncertainty.* Elements of the area of uncertainty may, or may not, belong to X.

The approximation operators can also be considered using membership functions. It is possible to define a rough set membership function as presented in [Paw96]. The rough membership function μ_X associates to an element $x \in \mathcal{U}$ a value between 0 and 1 as follows:

$$\mu_X(x) = \frac{\text{card}(X \cap [x]_\theta)}{\text{card}([x]_\theta)}$$

This way,

- $\mu_X(x) = 1$ iff $x \in \underline{\mathcal{A}_\theta}(X)$,
- $\mu_X(x) = 0$ iff $x \notin \overline{\mathcal{A}_\theta}(X)$ and
- $0 < \mu_X(x) < 1$ iff $x \in \overline{\mathcal{A}_\theta}(X) \setminus \underline{\mathcal{A}_\theta}(X)$.

For any sets $X, Y \subseteq \mathcal{U}$, given $\mathcal{A} = (\mathcal{U}, \theta)$, the following properties of approximations hold:

$$\underline{\mathcal{A}_\theta}(X) \subseteq X \subseteq \overline{\mathcal{A}_\theta}(X) \tag{3.1}$$

$$\underline{\mathcal{A}_\theta}(X \cap Y) = \underline{\mathcal{A}_\theta}(X) \cap \underline{\mathcal{A}_\theta}(Y) \tag{3.2}$$

$$\overline{\mathcal{A}_\theta}(X \cap Y) \subseteq \overline{\mathcal{A}_\theta}(X) \cap \overline{\mathcal{A}_\theta}(Y) \tag{3.3}$$

$$\underline{\mathcal{A}_\theta}(X \cup Y) \supseteq \underline{\mathcal{A}_\theta}(X) \cup \underline{\mathcal{A}_\theta}(Y) \tag{3.4}$$

$$\overline{\mathcal{A}_\theta}(X \cup Y) = \overline{\mathcal{A}_\theta}(X) \cup \overline{\mathcal{A}_\theta}(Y) \tag{3.5}$$

$$\underline{\mathcal{A}_\theta}(X \setminus Y) = \underline{\mathcal{A}_\theta}(X) \setminus \underline{\mathcal{A}_\theta}(Y) \tag{3.6}$$

$$\overline{\mathcal{A}_\theta}(X \setminus Y) \subseteq \overline{\mathcal{A}_\theta}(X) \setminus \overline{\mathcal{A}_\theta}(Y) \tag{3.7}$$

$$X \subseteq Y \Longrightarrow \underline{\mathcal{A}_\theta}(X) \subseteq \underline{\mathcal{A}_\theta}(Y) \tag{3.8}$$

$$X \subseteq Y \Longrightarrow \overline{\mathcal{A}_\theta}(X) \subseteq \overline{\mathcal{A}_\theta}(Y) \tag{3.9}$$

Proofs of the properties above can be found in [Paw91].

Rough sets are defined using lower and upper approximations. Let $\sim$ be an equivalence relation defined as:

$$\forall X, Y \subseteq \mathcal{U},\ X \sim Y \text{ if } \underline{\mathcal{A}_\theta}(X) = \underline{\mathcal{A}_\theta}(Y) \text{ and } \overline{\mathcal{A}_\theta}(X) = \overline{\mathcal{A}_\theta}(Y)$$

then, $\sim$ is an equivalence relation on the set of subsets of $\mathcal{U}$, i. e. the power set $2^{\mathcal{U}}$. The equivalence classes of the equivalence relation $\sim$ are *rough sets.* The set of all rough sets in the approximation space $(\mathcal{U}, \theta)$, denoted $\mathcal{R}_\theta(\mathcal{U})$, is the quotient set $2^{\mathcal{U}}/\sim$.

A rough set $R \in \mathcal{R}_\theta(\mathcal{U})$ is uniquely determined by the pair $(\underline{\mathcal{A}_\theta}(X), \overline{\mathcal{A}_\theta}(X))$. Here the set X is called the *representative* of the rough set R. The representative of a rough set is the set that is being approximated in the approximation space $(\mathcal{U}, \theta)$.

A rough set $(\underline{\mathcal{A}_\theta}(X), \overline{\mathcal{A}_\theta}(X))$ is said to be *crisp*, or *definable*, if $\overline{\mathcal{A}_\theta}(X) \setminus \underline{\mathcal{A}_\theta}(X) = \emptyset$. It is said to be *rough*, or *not definable*, otherwise.

A set $Y \subseteq \mathcal{U}$ is member of the rough set $R = (\underline{\mathcal{A}_\theta}(X), \overline{\mathcal{A}_\theta}(X))$, denoted $Y \in R$, if $\underline{\mathcal{A}_\theta}(Y) = \underline{\mathcal{A}_\theta}(X)$ and $\overline{\mathcal{A}_\theta}(Y) = \overline{\mathcal{A}_\theta}(X)$, hence, any member of R can be used as a representative for the rough set R as stated by the following equality:

$$R = (\underline{\mathcal{A}_\theta}(X), \overline{\mathcal{A}_\theta}(X)) = (\underline{\mathcal{A}_\theta}(Y), \overline{\mathcal{A}_\theta}(Y))$$

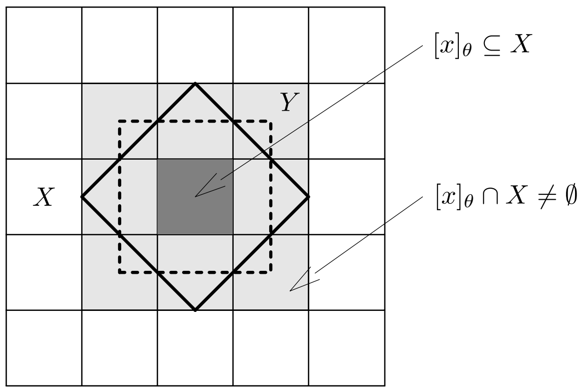


Figure 3.1. An example rough set. The dark gray area is the lower approximation; the light gray is the upper approximation.

Figure 3.1 shows an example rough set $R = (\underline{\mathcal{A}_\theta}(X), \overline{\mathcal{A}_\theta}(X))$ with representative X. As shown in the figure, since $\underline{\mathcal{A}_\theta}(X) = \underline{\mathcal{A}_\theta}(Y)$ and $\overline{\mathcal{A}_\theta}(X) = \overline{\mathcal{A}_\theta}(Y)$, Y is member of the rough set R, hence, both X and Y can be chosen as representative for the rough set R.

3.2 Truth Functionality and Rough Sets

There is an issue on truth-functionality of set theoretic operators and approximations.

For recall, an operator is said to be truth functional when the result of the application of the operator on the operands can be figured out solely on the basis of the operands. This is not the case for intersection and union of upper and lower approximations.

Let $(\underline{\mathcal{A}_\theta}(X), \overline{\mathcal{A}_\theta}(X))$ and $(\underline{\mathcal{A}_\theta}(Y), \overline{\mathcal{A}_\theta}(Y))$ be two rough sets with representatives X and Y respectively, then it may be that $\overline{\mathcal{A}_\theta}(X) \cap \overline{\mathcal{A}_\theta}(Y) \neq \overline{\mathcal{A}_\theta}(X \cap Y)$ and $\underline{\mathcal{A}_\theta}(X) \cup \underline{\mathcal{A}_\theta}(Y) \neq \underline{\mathcal{A}_\theta}(X \cup Y)$ according to properties (3.3) and (3.4), respectively. One cannot compute the intersection of upper approximations and the union of lower approximations based solely on the operands. It is necessary to consider the equivalence relation together with the representatives that are involved.

Let us simplify the notations. If the approximation space $\mathcal{A} = (\mathcal{U}, \theta)$ is understood, the notations for lower and upper approximations of the set X, i. e. $\underline{\mathcal{A}_\theta}(X)$ and $\overline{\mathcal{A}_\theta}(X)$, may be simplified to $\overline{X}$ and $\underline{X}$, respectively.

As a consequence of non-truth-functionality, set theoretic operators on $\mathcal{R}_\theta(\mathcal{U})$, such as union and intersection, cannot be defined component-wise using standard set operators. That is, it may be that:

$$(\underline{X}, \overline{X}) \cap (\underline{Y}, \overline{Y}) \neq (\underline{X \cap Y}, \overline{X \cap Y})$$

$$(\underline{X}, \overline{X}) \cup (\underline{Y}, \overline{Y}) \neq (\underline{X \cup Y}, \overline{X \cup Y}).$$

Hence, intersection and union of rough sets are non-truth-functional.

Let us illustrate this with some figures. Figure 3.2(a) shows two rough sets $(\underline{X}, \overline{X})$ and $(\underline{Y}, \overline{Y})$ with representatives X and Y respectively. In the figure, we have $X \cap Y = \emptyset$, however, the upper approximations $\overline{X}$ and $\overline{Y}$ intersect. In figure 3.2(b), the lower approximation of X is shown in dark gray, the lower approximation of Y is empty. However, as shown in the figure by the dashed arrow, the union of X and Y contains one more equivalence class.

Below, we look at some alternative views of rough sets.

3.2.1 Families of Subsets

Yao in [Yao96] argues that rough sets can be specified without the use of representatives. In this case, a rough set can be written as the pair (L, U), where L and U are unions of equivalence classes of $\mathcal{U}/\theta$. Here, we only know that $\underline{X} = L$ and $\overline{X} = U$ for some unknown set X, hence, (L, U) is specified as follows:

$$(L, U) = \{X \in 2^{\mathcal{U}} | \underline{\mathcal{A}_\theta}(X) = L \text{ and } \overline{\mathcal{A}_\theta}(X) = U\}$$

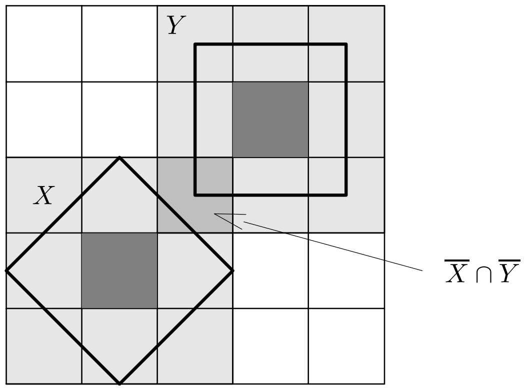
Here, (L, U) defines a family of subsets of the universe.

In this setting, an intersection operator, denoted $\sqcap$, and a union operator, denoted $\sqcup$, may be defined on the rough sets (L, U) and (L', U') as:

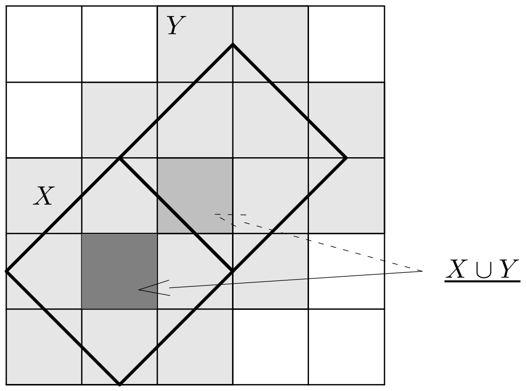
$$(L, U) \sqcap (L', U') = \{X \in 2^{\mathcal{U}} | \underline{\mathcal{A}_\theta}(X) = L \cap L' \text{ and } \overline{\mathcal{A}_\theta}(X) = U \cap U'\}$$

$$(L, U) \sqcup (L', U') = \{X \in 2^{\mathcal{U}} | \underline{\mathcal{A}_\theta}(X) = L \cup L' \text{ and } \overline{\mathcal{A}_\theta}(X) = U \cup U'\}$$

Although there is no representatives used in this interpretation of the rough sets, the intersection and union operators are still not truth functional in the sense that



(a) Non-truth-functionality of the upper approximation operator.



(b) Non truth-functionality of the lower approximation operator.

Figure 3.2. Examples illustrating the non truth-functionality of the lower and the upper approximation operators.

operations on the set of rough sets in this view are not defined using all members of the rough sets [YL96].

3.2.2 Interval Sets

Recall the definition of interval set of Section 2.7.

For two interval sets $\mathcal{I} = [I, I']$ and $\mathcal{J} = [J, J']$, the following truth-functional operations on interval sets are defined:

$$\mathcal{I} \sqcap \mathcal{J} = [I \cap J, I' \cap J'] \tag{3.10}$$

$$\mathcal{I} \sqcup \mathcal{J} = [I \cup J, I' \cup J'] \tag{3.11}$$

$$\mathcal{I} \setminus \mathcal{J} = [I - J', I' - J] \tag{3.12}$$

According to Yao [Yao93], rough sets provide a way to construct interval sets. Provided a finite universe $\mathcal{U}$, an equivalence relation θ, and a set X we wish to approximate, an interval set can be constructed as $\left[\underline{\mathcal{A}_\theta}(X), \overline{\mathcal{A}_\theta}(X)\right]$.

The reverse process, however, is not straightforward. That is, given a finite universe $\mathcal{U}$ and an interval set $[I, I']$ where $I, I' \subseteq 2^{\mathcal{U}}$, there might be more than one approximation space $\mathcal{A} = (\mathcal{U}, \theta)$ such that $I = \underline{\mathcal{A}_\theta}(X)$ and $I' = \overline{\mathcal{A}_\theta}(X)$ for some X. Hence, we do not have enough information to reconstruct the initial rough set.

3.2.3 Restriction of Approximations

Here, we present an alternative based on restriction of approximations. Instead of considering all subsets of the universe $2^{\mathcal{U}}$ in the approximation space $(\mathcal{U}, \theta)$, only some particular subsets $\mathcal{R} \subset 2^{\mathcal{U}}$ are considered. The idea is to choose $\mathcal{R}$ in such a way that truth-functionality of the intersection and union operators hold, i. e. $\forall X, Y \in \mathcal{R}$, $\underline{\mathcal{A}_\theta}(X \cup Y) = \underline{\mathcal{A}_\theta}(X) \cup \underline{\mathcal{A}_\theta}(Y)$ and $\overline{\mathcal{A}_\theta}(X \cap Y) = \overline{\mathcal{A}_\theta}(X) \cup \overline{\mathcal{A}_\theta}(Y)$ hold. To simplify the terminology, in this case, we say that $\mathcal{R}$ is a set of *truth-functional representatives.*

Below, we present a way to achieve this as described in [Yao93]. The method is based on a uniform set of representatives that has been introduced by Gehrke et al. in [GW92]. We also extend this method later to fit our need to represent spatial uncertainty.

Uniform Set of Representatives

For recall, a set of class representatives of a quotient set is a set that contains exactly one element from each equivalence class of the quotient set.

For any equivalence class $E \in \mathcal{U}/\theta$ with at least two elements, we define a set A_E such that $\emptyset \subset A_E \subset E$ and we let $A = \bigcup_{E \in \mathcal{U}/\theta} A_E$. We call A the *template set.* If the equivalence class E contains exactly one element we set $A_E = \emptyset$. Furthermore, we let $\mathcal{R}_A$ denote the set defined as:

$$\mathcal{R}_A = \{X_A | X_A = \underline{X} \cup (\overline{X} \cap A), \forall X \subseteq \mathcal{U}\}$$

The set $\mathcal{R}_A$ is a set of class representatives of $2^{\mathcal{U}}/\sim$. This is the case since $X_A \neq Y_A \Longrightarrow X_A \not\sim Y_A$ as stated in the following lemma.

Lemma 3.1. For any pair of elements $X_A, Y_A \in \mathcal{R}_A$, if $X_A \neq Y_A$ then $X_A \not\sim Y_A$.

Proof.

$$X_A \neq Y_A \Longrightarrow \underline{X} \cup (\overline{X} \cap A) \neq \underline{Y} \cup (\overline{Y} \cap A) \Longrightarrow \begin{cases} \underline{X} \neq \underline{Y} \\ \vee \\ (\overline{X} \cap A) \neq (\overline{Y} \cap A) \end{cases}$$

because of the construction of A. Thus, either $\underline{X} \neq \underline{Y}$ which implies $\underline{X_A} \neq \underline{Y_A}$, or, $(\overline{X} \cap A) \neq (\overline{Y} \cap A)$. For the latter case to be true it must be the case that at least one element of A belongs to $\overline{X}$ but does not belong to $\overline{Y}$, or, conversely, at least one element of A belongs to $\overline{Y}$ and does not belong to $\overline{X}$. In both cases we have that $\overline{X_A} \neq \overline{Y_A}$. Hence, X_A and Y_A cannot belong to the same equivalence class of $2^{\mathcal{U}}/\sim$, i. e. $X_A \not\sim Y_A$. ■

As a consequence, to each subset $X \subseteq \mathcal{U}$, given a fixed set A defined as above, corresponds a unique class representative $X_A \in \mathcal{R}_A$ such that $X \sim X_A$.

Now, let us look at the truth-functionality issue.

Lemma 3.2. Let A be a template set defined as above, then $\forall X_A, Y_A \in \mathcal{R}_A$, the following equalities hold:

$$\underline{\mathcal{A}_\theta}(X_A \cup Y_A) = \underline{\mathcal{A}_\theta}(X_A) \cup \underline{\mathcal{A}_\theta}(Y_A) \tag{3.13}$$

$$\overline{\mathcal{A}_\theta}(X_A \cap Y_A) = \overline{\mathcal{A}_\theta}(X_A) \cap \overline{\mathcal{A}_\theta}(Y_A) \tag{3.14}$$

Proof. $\forall E \in \mathcal{U}/\theta$ and $X \subseteq \mathcal{U}$ we define $X_E = X \cap E$. Since $X = \bigcup_{E \in \mathcal{U}/\theta} X_E$ and $\bigcap_{E \in \mathcal{U}/\theta} X_E = \emptyset$, it suffices to show that both equalities 3.13 and 3.14 hold for $(X_E)_A$ and $(Y_E)_A$, that is:

$$\underline{(X_E)_A \cup (Y_E)_A} = \underline{(X_E)_A} \cup \underline{(Y_E)_A}$$

$$\overline{(X_E)_A \cap (Y_E)_A} = \overline{(X_E)_A} \cap \overline{(Y_E)_A}$$

In the case when $E \subseteq X$ or $E \subseteq Y$, either $(X_E)_A = E$ or $(X_E)_A = E$, in both cases we have that:

$$\underline{(X_E)_A \cup (Y_E)_A} = \underline{(X_E)_A} \cup \underline{(Y_E)_A} = E$$

$$\overline{(X_E)_A \cap (Y_E)_A} = \overline{(X_E)_A} \cap \overline{(Y_E)_A} = E$$

Let us consider the case when E intersects both X and Y and $E \not\subset X$ and $E \not\subset Y$, then, $\underline{X_E} = \underline{Y_E} = \emptyset$ and $\overline{X_E} \cap A_E = \overline{Y_E} \cap A_E = A_E$, hence, $(X_E)_A = (Y_E)_A = A_E$ and both equalities 3.13 and 3.14 hold. ■

Hence, if we restrict our domain to the set $\mathcal{R}_A$, then, intersection and union of approximations are truth-functional. As a consequence, intersection and union operations can be defined component-wise on the rough sets $(\underline{X}, \overline{X})$ and $(\underline{Y}, \overline{Y})$ when $X, Y \in \mathcal{R}_A$ as follows:

$$(\underline{X}, \overline{X}) \cap (\underline{Y}, \overline{Y}) = (\underline{X} \cap \underline{Y}, \overline{X} \cap \overline{Y})$$

$$(\underline{X}, \overline{X}) \cup (\underline{Y}, \overline{Y}) = (\underline{X} \cup \underline{Y}, \overline{X} \cup \overline{Y})$$

Restricted approximations may be used as described below.

Suppose we wish to approximate sets $X, Y \subseteq \mathcal{U}$ in an approximation space $(\mathcal{U}, \theta)$. We first define a template set A which will impose the representatives of $\mathcal{R}_A$ that are to be used. Next, we substitute sets X and Y by their corresponding representatives X_A and Y_A in $\mathcal{R}_A$, i. e. $X \sim X_A$ and $Y \sim Y_A$. In any subsequent operation, X_A and Y_A are used instead of X and Y.

Substituting arbitrary subsets X by their corresponding uniform representatives X_A is admissible with respect to a given equivalence relation $\sim$, however, sometimes we wish the substituent of a set X to satisfy an additional condition beside the equivalence $X \sim X_A$. An example of such a condition may be that the substituted set X and the substituent set X_A do not differ in too many elements, i. e. they have as many elements in common as possible. Obviously, this can be achieved if we succeed in growing the size of the set of truth-functional representatives $\mathcal{R}$.

Let us look at the example of Figure 3.3. In the figure, the template set A is shown as light gray filled circles. Figure 3.3(a) shows the set X, shown as a gray filled square, that is to be substituted. The class representative X_A that will substitute X is shown in Figure 3.3(b) as a set of black filled circles and a black filled square. The latter example shows that a class representative X_A of a substituted set X can differ in relatively many elements. In fact, it may happen that they do not have any elements in common even though $X \sim X_A$.

Below, we introduce a modified version of uniform set of representatives in which we do not allow substituted sets and their corresponding substituents to differ in more than $2 \cdot |\mathcal{U}/\theta|$ elements.

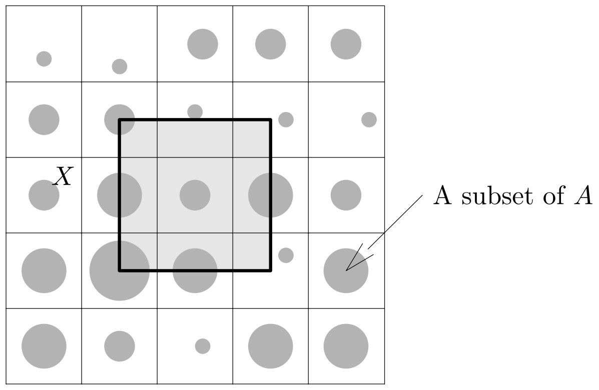
Extended Uniform Set of Representatives

Let A and B be two sets of class representatives from which we eliminate elements that are class representatives of singleton sets of $\mathcal{U}/\theta$. Moreover, sets A and B are defined such that $A \cap B = \emptyset$. Let $\mathcal{R}_{AB}$ denote the set defined as:

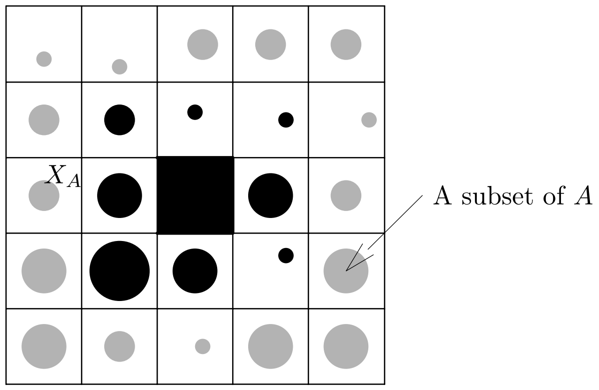
$$\mathcal{R}_{AB} = \Big\{X_{AB} | X_{AB} = \big(X \cup (\overline{X} \cap A)\big) \setminus \big((\overline{X} \setminus \underline{X}) \cap B\big), \forall X \subseteq \mathcal{U}\Big\}$$

Here too, sets A and B are called template sets.

A set X_{AB} defined as above may substitute the set X since $X \sim X_{AB}$. Moreover, by its construction, if an equivalence class $E \in \mathcal{U}/\theta$ intersects X such that $E \not\subset X$, then, $X \cap E$ and $X_{AB} \cap E$ may differ at most at two elements. That is, $X \cap E$ might not contain $A \cap E$ but might contain $B \cap E$, X_{AB}, on the other hand, contains $A \cap E$ but does not contain $B \cap E$. As a consequence, the substituted set and the substituent differ at most at $2 \cdot |\mathcal{U}/\theta|$ elements.



(a) The initial sets X.



(b) The class representative X_A

Figure 3.3. Substituted and substituent sets.

Lemma 3.3. Let A and B be two template sets defined as above, then for all X_{AB}, Y_{AB} members of $\mathcal{R}_{AB}$, the following equalities hold:

$$\underline{X_{AB} \cup Y_{AB}} = \underline{X_{AB}} \cup \underline{Y_{AB}} \tag{3.15}$$

$$\overline{X_{AB} \cap Y_{AB}} = \overline{X_{AB}} \cap \overline{Y_{AB}} \tag{3.16}$$

Proof. Here too, we pose $X_E = X \cap E$, where $E \in \mathcal{U}/\theta$, and show that both equalities 3.15 and 3.16 hold for $(X_E)_{AB}$ and $(Y_E)_{AB}$. In the case when $E \subseteq X$ or

$E \subseteq Y$, either $(X_E)_{AB} = E$ or $(X_E)_{AB} = E$, in both cases we have that:

$$\underline{(X_E)_{AB} \cup (Y_E)_{AB}} = \underline{(X_E)_{AB}} \cup \underline{(Y_E)_{AB}} = E$$

$$\overline{(X_E)_{AB} \cap (Y_E)_{AB}} = \overline{(X_E)_{AB}} \cap \overline{(Y_E)_{AB}} = E$$

Let us consider the case when E intersects both X and Y and $E \not\subset X$ and $E \not\subset Y$. Then, by their construction, $\underline{(X_E)_{AB}}$ and $\underline{(Y_E)_{AB}}$ have an element in common which is $(A \cap E)$ and an element $(B \cap E)$ which is missing from both $\underline{(X_E)_{AB}}$ and $\underline{(Y_E)_{AB}}$. Hence, both equalities 3.15 and 3.16 hold. ∎

Hence, intersection and union of approximations of elements of $\mathcal{R}_{AB}$ are truth-functional.

An example element X_{AB} of $\mathcal{R}_{AB}$ is shown in Figure 3.4. In the figure, a black dot represents an element of the template set A and a gray dot represents an element of the template set B.

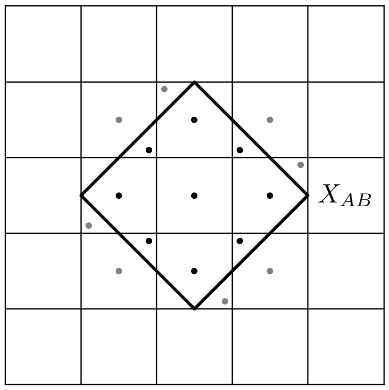


Figure 3.4. An example element X_{AB}

As previously, if we restrict our domain to the set $\mathcal{R}_{AB}$, then, for rough sets $(\underline{X}, \overline{X})$ and $(\underline{Y}, \overline{Y})$ where the representatives are such that $X, Y \in \mathcal{R}_{AB}$, we have that:

$$(\underline{X}, \overline{X}) \cap (\underline{Y}, \overline{Y}) = (\underline{X} \cap \underline{Y}, \overline{X} \cap \overline{Y})$$

$$(\underline{X}, \overline{X}) \cup (\underline{Y}, \overline{Y}) = (\underline{X} \cup \underline{Y}, \overline{X} \cup \overline{Y})$$

There is an interesting issue concerning the set of truth-functional representatives $\mathcal{R}_{AB}$: the complement of an element of $\mathcal{R}_{AB}$ is not in $\mathcal{R}_{AB}$. In fact, the complement of an element X_{AB} in $\mathcal{U}$, i. e. $(\mathcal{U} \setminus X_{AB})$, is an element of $\mathcal{R}_{BA}$. Hence, sets A and B play a symmetric role.

3.3 Fuzzy Sets

Let $\mathcal{U}$ be a universe of discourse. A standard set A in $\mathcal{U}$ may be defined as a set of ordered pairs $A = \{(x, I_A(x)) | x \in \mathcal{U}\}$, where $I_A(x) \in \{0, 1\}$ is an *indicator function*. For a given data point x, $I_A(x)$ indicates the membership of x in the set A, 0 for x non-member of the set A and 1 for x member of the set A.

L. A. Zadeh has introduced fuzzy sets in [Zad65]. In a fuzzy set it is possible to represent the degree of membership of a data point to a given set by a value ranging from 0 to 1: a *fuzzy set* A in $\mathcal{U}$ is a set of ordered pairs $A = \{(x, \mu_A(x)) | x \in \mathcal{U}\}$, defined by a *membership function* $0 \leq \mu_A(x) \leq 1$.

Basic set operations extend to fuzzy sets. Let A and B be fuzzy sets defined as $A = \{(x, \mu_A(x)) | x \in \mathcal{U}\}$ and $B = \{(x, \mu_B(x)) | x \in \mathcal{U}\}$. The union and intersection of fuzzy sets A and B are commonly defined as:

$$A \cup B = \{(x, \mu_{A \cup B}(x)) | x \in \mathcal{U}\}, \text{where } \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)),$$

$$A \cap B = \{(x, \mu_{A \cap B}(x)) | x \in \mathcal{U}\}, \text{where } \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)).$$

The complement of a fuzzy set $A = \{(x, \mu_A(x)) | x \in \mathcal{U}\}$ in $\mathcal{U}$, denoted $\neg A$, is defined as $\neg A = \{(x, \mu_{\neg A}(x)) | x \in \mathcal{U}\}$, where $\mu_{\neg A}(x) = 1 - \mu_A(x)$.

The *support* of a fuzzy set A is the crisp set containing all non-zero members of A. The support of a set A is denoted Support(A). An element x belongs to Support(A) if $\mu_A(x) > 0$.

According to [Yao97], it may be desirable to explicitly establish a connection between fuzzy sets and crisp sets. In this case one might use instead another representation of fuzzy sets that is based on crisp sets. Formally, given $\alpha \in [0, 1]$, an *α-cut* or *α-level set*, of a fuzzy set A is defined as:

$$A_\alpha = \{x \in \mathcal{U} | \mu_A(x) \geq \alpha\},$$

a strong α-cut is defined as:

$$A_{\alpha^+} = \{x \in \mathcal{U} | \mu_A(x) > \alpha\}.$$

Fuzzy sets can be reconstructed from their α-cuts, i. e. it is possible to retrieve the membership function μ_A based solely on the α-level sets A_α:

$$\mu_A(x) = \sup\{\alpha | x \in A_\alpha\}$$

3.4 Rough Fuzzy Sets

Dubois and Prade [DP92] have proposed the concept of *rough fuzzy set*, which is a rough approximation of a fuzzy set. Rough fuzzy sets are based on approximation spaces $\mathcal{A} = (\mathcal{U}, \theta)$. Let $E = [x]_\theta \in \mathcal{U}/\theta$, we wish to approximate a fuzzy set $X \subseteq \mathcal{U}$ in the approximation space $\mathcal{A}$. The lower and upper approximations $\underline{\mathcal{A}_\theta}(X)$

and $\overline{\mathcal{A}_\theta}(X)$ of a fuzzy set X are fuzzy sets of $\mathcal{U}/\theta$ with membership functions $\mu_{\underline{\mathcal{A}_\theta}(X)}, \mu_{\overline{\mathcal{A}_\theta}(X)} : \mathcal{U}/\theta \to [0,1]$. The membership functions of the lower and upper approximations of X are defined respectively as:

$$\mu_{\underline{\mathcal{A}_\theta}(X)}(E) = \inf\{\mu_X(x) | E = [x]_\theta\}$$

$$\mu_{\overline{\mathcal{A}_\theta}(X)}(E) = \sup\{\mu_X(x) | E = [x]_\theta\}$$

Given a fuzzy set X, the pair $(\mu_{\underline{\mathcal{A}_\theta}(X)}, \mu_{\overline{\mathcal{A}_\theta}(X)})$ is called a *rough fuzzy set*. One can also extend the membership functions $\mu_{\underline{\mathcal{A}_\theta}(X)}$ and $\mu_{\overline{\mathcal{A}_\theta}(X)}$ to be defined over the universe $\mathcal{U}$ using the extension principle introduced by Zadeh [Zad65]. This way, the membership functions are defined as:

$$\mu_{\underline{\mathcal{A}_\theta}(X)}, \mu_{\overline{\mathcal{A}_\theta}(X)} : \mathcal{U} \to [0,1]$$

$$\mu_{\underline{\mathcal{A}_\theta}(X)}(x) = \inf\{\mu_X(y) | y \in [x]_\theta\}$$

$$\mu_{\overline{\mathcal{A}_\theta}(X)}(x) = \sup\{\mu_X(y) | y \in [x]_\theta\}$$

Rough fuzzy sets are able to express uncertainty due to indiscernibility as well as uncertainty due to vagueness.

Given an approximation space $(\mathcal{U}, \theta)$ and fuzzy sets $X, Y \subseteq \mathcal{U}$, $\forall E \in \mathcal{U}/\theta$ and $\forall x \in \mathcal{U}$, the following properties of approximations of rough fuzzy sets may be deduced from the properties of rough sets presented in Section 3.1 (see also [Thi98]):

$$\mu_{\underline{\mathcal{A}_\theta}(X)}(E) \leq \mu_X(E) \leq \mu_{\overline{\mathcal{A}_\theta}(X)}(E) \tag{3.17}$$

$$\mu_{\underline{\mathcal{A}_\theta}(X \cap Y)}(E) = \min(\mu_{\underline{\mathcal{A}_\theta}(X)}(E), \mu_{\underline{\mathcal{A}_\theta}(Y)}(E)) \tag{3.18}$$

$$\mu_{\overline{\mathcal{A}_\theta}(X \cap Y)}(E) <= \min(\mu_{\overline{\mathcal{A}_\theta}(X)}(E), \mu_{\overline{\mathcal{A}_\theta}(Y)}(E)) \tag{3.19}$$

$$\mu_{\underline{\mathcal{A}_\theta}(X \cup Y)}(E) >= \max(\mu_{\underline{\mathcal{A}_\theta}(X)}(E), \mu_{\underline{\mathcal{A}_\theta}(Y)}(E)) \tag{3.20}$$

$$\mu_{\overline{\mathcal{A}_\theta}(X \cup Y)}(E) = \max(\mu_{\overline{\mathcal{A}_\theta}(X)}(E), \mu_{\overline{\mathcal{A}_\theta}(Y)}(E)) \tag{3.21}$$

$$\mu_X(x) \leq \mu_Y(x) \Longrightarrow \mu_{\underline{\mathcal{A}_\theta}(X)}(E) \leq \mu_{\underline{\mathcal{A}_\theta}(Y)}(E) \tag{3.22}$$

$$\mu_X(x) \leq \mu_Y(x) \Longrightarrow \mu_{\overline{\mathcal{A}_\theta}(X)}(E) \leq \mu_{\overline{\mathcal{A}_\theta}(Y)}(E) \tag{3.23}$$

See [Pol02] for proofs of the properties above.

If the approximation space $\mathcal{A} = (\mathcal{U}, \theta)$ is understood, the notations for lower and upper approximations, $\mu_{\underline{\mathcal{A}_\theta}(X)}$ and $\mu_{\overline{\mathcal{A}_\theta}(X)}$, of a fuzzy set X, may be simplified to $\mu_{\underline{X}}$ and $\mu_{\overline{X}}$, respectively.

Intersection and union of rough fuzzy sets are non-truth-functional according to properties 3.19 and 3.20. However, restriction of approximations of Section 3.2.3 also extends to rough-fuzzy sets. Hence, truth-functional intersection and union operations may be defined on rough-fuzzy sets as shown below.

The definition of extended uniform set representatives may be adapted to hold for rough fuzzy sets. This time, we wish to build a set of fuzzy set representatives

$\mathcal{F}_{AB}$ such that they support truth-functional intersection and union operations. The fuzzy sets A and B, with membership functions μ_A and μ_B respectively, are such that Support(A) and Support(B) are two disjoint set of class representatives of $\mathcal{U}/\theta$.

We wish to achieve that if $X, Y \in \mathcal{F}_{AB}$ then the following equalities hold:

$$\mu_{\overline{X \cap Y}}(E) = \min(\mu_{\overline{X}}(E), \mu_{\overline{Y}}(E))$$

$$\mu_{\underline{X \cup Y}}(E) = \max(\mu_{\underline{X}}(E), \mu_{\underline{Y}}(E))$$

This turns out to hold only if:

$$\forall \alpha \in \mathbb{R}, X_\alpha, Y_\alpha \in \mathcal{R}_{A_\alpha B_\alpha}$$

for some fixed fuzzy sets A and B.

3.5 Conclusion

As seen in this chapter, intersection and union of approximations are non-truth-functional. We point out two main issues caused by non-truth-functionality of rough sets in our spatial information system.

The first issue is related to unknown set representatives. For instance, assume that two unknown subsets of the universe X and Y have been approximated in some approximation space. Moreover, the only information at hand is the lower and upper approximations of the sets in question, i. e. the rough sets (L, U) and (L', U'), then we cannot compute the union and intersection of these two rough sets.

The second issue is related to measuring areas of uncertainty. Consider that the elements of the universe are now polygons, representing some spatial information, with an area. In this case, since $\overline{X \cap Y} \subseteq \overline{X} \cap \overline{Y}$, the following inequality holds:

$$\text{area}(\overline{X}) \cap \text{area}(\overline{Y}) \geq \text{area}(\overline{X \cap Y})$$

Hence, performing intersection of upper approximations leads to overestimating the area of intersection.

We have shown that it is possible to avoid the truth-functionality issues by changing representatives. The drawback is, of course, that we do not approximate the initial set but some substituent. Clearly, this leads to introducing some more imprecision in the data set.

There is, however, another way we may look at the problem. The extended uniform set of representatives may be seen as a property. The fact that two sets X and Y belong to $\mathcal{R}_{AB}$, for some disjoint class representatives A and B, makes intersection and union operations on X and Y truth-functional. Hence, given a set of subsets X_i we wish to approximate, we may look for the set of representatives A and B such that $X_i \in \mathcal{R}_{AB}$ for all i.

Hence, we may be given pairs of upper and lower approximations (L_i, U_i) with the claim that they are the result of some approximations $L_i = \underline{X_i}$ and $U_i = \overline{X_i}$ where, for all i, $X_i \in \mathcal{R}_{AB}$. In this case, we may combine the pairs of approximations with intersection and union operations. As a consequence, computing area of polygons, or cardinality of sets, may be done exactly:

$$\text{area}(U_i) \cap \text{area}(U_j) = \text{area}(U_i \cap U_j)$$

Chapter 4

Spatial and Attribute Uncertainty

In this chapter we extend the query language by Marek et al. [MT99] in a way that allows us to represent indiscernibility and vagueness. To represent indiscernibility we use rough set theory and to represent vagueness we use fuzzy set theory.

The query language presented in [MT99] is based on the concept of information system. Our query language, on the other hand, is based on multi-valued information systems.

We first recall the concept of multi-valued information systems, then Section 4.2 shows how we extend this concept to support approximation spaces. Section 4.3 and 4.4 present a novel approach to perform classifications. Finally, in Section 4.5, we show how to extend our model to support vagueness.

4.1 Multi-valued Information Systems

Let $\mathcal{U}$ be a finite set of elements called the universe and $\mathcal{AT}$ a non-empty finite set of attributes $a \in \mathcal{AT}$, such that $a : \mathcal{U} \to \mathcal{V}_a$. The set $\mathcal{V}_a$ is called the *range* of the attribute a. For an element $x \in \mathcal{U}$ and an attribute $a \in \mathcal{AT}$, the pair $(x, a(x)) \in \mathcal{U} \times \mathcal{V}_a$ indicates that x has the attribute value $a(x)$. The pair $(\mathcal{U}, \mathcal{AT})$ is called an *information system* [DGO01] and is often referred to as a single-valued information system. In a single-valued information system, attributes $a \in \mathcal{AT}$ map elements $x \in \mathcal{U}$ to a single attribute value $v = a(x)$ in the range $\mathcal{V}_a$.

A multi-valued information system is a generalization of the idea of a single-valued information system. In a multi-valued information system, attribute functions are allowed to map elements to sets of attribute values [KR96, DGO01] (see also [DG00]). More formally, we allow multi-valued attributes a such that $a : \mathcal{U} \to 2^{\mathcal{V}_a}$. A subset $a(x) \subseteq \mathcal{V}_a$ may also be referred to as an *attribute value*.

Since a may map x to more than one attribute value at a time we need an interpretation for $a(x) = V$. Düntsch et al. in [DGO01] propose the following ones:

(**CEX**) V is interpreted conjunctively and exhaustively, i. e. all elements of V and only those elements are attributes of element x. As an example, consider the attribute a, *cooperates with*, where $a(\text{'Karim'}) = \{\text{'Johannes'}, \text{'Ola'}\}$. In this interpretation, 'Karim' cooperates with 'Johannes' and 'Ola' and no one else.

(**CNE**) V is interpreted conjunctively and non-exhaustively, i. e. all elements of V and possibly others are attributes of element x. Using the previous example, in this case, 'Karim' cooperates with 'Johannes' and 'Ola' and possibly others.

(**DEX**) V is interpreted disjunctively and exclusively, i. e. one and only one element of V is attribute of element x but it is not known which one. Here, 'Karim' cooperates with 'Johannes' or 'Ola' but not both.

(**DNE**) V is interpreted disjunctively and non-exclusively, i. e. at least one element of V and possibly more are attributes of element x. Finally, in this case, 'Karim' cooperates with 'Johannes' or 'Ola' or both.

Subsequently, we consider the **CEX** interpretation. In this interpretation, the attribute value of an element of the universe may be composed of several attribute values. For instance, in our previous example where 'Karim' cooperates with 'Johannes' and 'Ola', both values 'Johannes' and 'Ola' compose the attribute values 'Karim' *cooperates with*. On the other hand, a disjunctive interpretations is suitable to represent the uncertainty in the value that an element of the universe maps to. For instance, 'Karim' cooperates with 'Johannes' or 'Ola' but we do not know which one of them.

Now, let us discuss how to relate elements of the universe to their attribute values.

In a multi-valued information system $(\mathcal{U}, \mathcal{AT})$, each attribute $a \in \mathcal{AT}$ implies a relation $R_a \subseteq \mathcal{U} \times \mathcal{V}_a$ by setting $x\, R_a\, v \Leftrightarrow v \in a(x)$. This way, each element of the information system can be associated a description by means of a subset $A \subseteq \mathcal{AT}$ of attributes, called the A-*description*, denoted $A(x)$. The tuple $A(x)$ is defined as $A(x) = \prod_{a \in A} a(x)$ such that $A(x)$ belongs to $\prod_{a \in A} 2^{\mathcal{V}_a}$. A tuple $A(x)$ may also be denoted as $\langle V \rangle_A$, where V is the value of $A(x)$, or just $\langle V \rangle$ if there is no risk for confusion.

A useful operation on tuples is the projection operation. The *projection* operator, denoted π_i, maps the i-th element V_i of a tuple as follows:

$$\begin{aligned} 2^{\mathcal{V}_1} \times \ldots \times 2^{\mathcal{V}_i} \ldots \times 2^{\mathcal{V}_n} &\xrightarrow{\pi_i} 2^{\mathcal{V}_i} \\ \langle V_1, \ldots, V_i, \ldots, V_n \rangle &\longmapsto V_i \end{aligned}$$

The projection operator π_i may be generalized for convenience to a projection operator π_B that projects a tuple $\langle V \rangle \subseteq \prod_{a \in A} 2^{\mathcal{V}_a}$ on the attributes of a set of attributes B such that $B \subseteq A$.

Subsequently, we refer to multi-valued information systems simply as information systems, a single-valued information system being a particular case of a multi-valued information system.

4.2 Information Systems and Approximation

In this thesis, we further extend multi-valued information systems $(\mathcal{U}, \mathcal{AT})$ by introducing an equivalence relation $\sim_a$, where $\sim_a \subseteq \mathcal{V}_a \times \mathcal{V}_a$, on the range $\mathcal{V}_a$ of each attribute $a \in \mathcal{AT}$. By setting equivalence relations on the attribute values of $\mathcal{V}_a$ we are able to model indiscernibility of attribute values. The only discernible elements are the equivalence classes of $\mathcal{V}_a/\sim_a$, hence, the name of equivalence classes play also the role of attribute values. For instance, if the equivalence class *blue* contains the attribute values 'sky blue' and 'light blue', then 'blue' is also considered as an attribute value.

In this information system, one cannot distinguish an attribute value $V \subseteq \mathcal{V}_a$, instead, the attribute value is approximated in the approximation space $(\mathcal{V}_a, \sim_a)$. If an element x of the universe has the attribute value $a(x) = V$, we say that the attribute value of x that is *discerned* is the rough set $(\underline{V}, \overline{V})$. Note that each individual element of the equivalence classes still remain represented in the information system. However, some users of the information system may only discern the equivalence classes.

Given an attribute value $V \subseteq \mathcal{V}_a$, the set $\underline{V}$ is the set of attribute values of $\mathcal{V}_a$ which *certainly* belong to V. $\overline{V}$ is the set of elements which *possibly* belong to V. The set $\mathcal{V}_a \setminus \overline{V}$ *certainly* belongs to the complement of V in $\mathcal{V}_a$, i. e. *certainly* does not belong to V. This interpretation extends to elements of the universe as shown below.

Let x be an element of the universe $\mathcal{U}$ such as $a(x) = V$, then:

- the attribute values of $\underline{a(x)}$ *certainly* belong to the attribute values of x. We say that $\underline{a(x)}$ are *certain* attribute values of x.
- the attribute values of $\overline{a(x)}$ *possibly* belong to the attribute values of x. We say that $\overline{a(x)}$ are *possible* attribute values of x.
- the attribute values of $\mathcal{V}_a \setminus \overline{a(x)}$ *certainly* do not belong to the attribute values of x.

Let us classify multi-valued information systems into *single-granular information systems* and *multi-granular information systems*. In a single-granular information system we restrict elements of the universe to have their attribute values in one and only one equivalence class, i. e. $a(x) = V/\sim_a$ is a singleton for every element x and attribute a. We do not have such a restriction in multi-granular information systems.

4.2.1 Indiscernibility of Attribute Values

Some attribute values may not be discernible from each other in the sense that they may have identical upper and lower approximations in the approximation spaces $(\mathcal{V}_a, \sim_a)$.

More formally, two attribute values $V, V' \subseteq \mathcal{V}_a$ are indiscernible from each other, denoted $V \approx_a V'$, if:

$$\left(\underline{\mathcal{A}_{\sim_a}}(V) = \underline{\mathcal{A}_{\sim_a}}(V')\right) \bigwedge \left(\overline{\mathcal{A}_{\sim_a}}(V) = \overline{\mathcal{A}_{\sim_a}}(V')\right)$$

It is easily seen that $\approx_a$ is an equivalence relation. The equivalence class of a subset $V \subseteq \mathcal{V}_a$ in $\approx_a$ is denoted $[V]_{\approx_a}$ and is an element of the quotient set $2^{\mathcal{V}_a} / \approx_a$.

Let us look at an example.

Example 4.1. An example spatial information system consists of a set of polygons. Each polygon has a number of properties determined by a multi-valued attribute `Colors`. Hence a given polygon may be characterized by several colors. The following set of attribute values, denoted $\mathcal{V}_c$, will be considered: lime green (lg), yellow green (yg), medium sea green (mg), sky blue (sb), light blue (lb), slate gray (sg), dark gray (dg) and white. For obvious reasons we impose the equivalence relation $\sim$ on the set of attribute values as:

$$\mathcal{V}_c / \sim = \Big\{ \underbrace{\{\text{lg}, \text{yg}, \text{mg}\}}_{\text{green}}, \underbrace{\{\text{sb}, \text{lb}\}}_{\text{blue}}, \underbrace{\{\text{sg}, \text{dg}\}}_{\text{gray}}, \{\text{white}\} \Big\}$$

As shown above, the equivalence classes of our example are named 'green', 'blue', 'gray' and 'white'. In this information system, only the equivalence classes representing the colors 'green', 'blue', 'gray' and 'white' are discernible.

Now, consider a polygon p_i with the attribute value `Colors`$(p_i) = \{\text{lg}, \text{mg}\} = V_1$, the lower approximation $\underline{V_1}$ in empty and $\overline{V_1}$ = green, hence, V_1 is not definable in our example approximation space. However, it can be approximated using the rough set $(\emptyset, \text{green})$. Hence, we can only state that the attribute values 'lime green', 'yellow green' and 'medium sea green' are possibly attribute values of p_i according to the interpretation of rough sets.

Assume now that there is a second polygon p_j where `Colors`$(p_j) = \{\text{yg}, \text{mg}\} = V_2$. Although V_1 and V_2 are different sets in the approximation space at hand, the attribute values V_1 and V_2 are indiscernible since the following hold:

$$\emptyset = \underline{V_1} = \underline{V_2} \subsetneq \overline{V_1} = \overline{V_2} = \text{green}$$

As a consequence, polygons p_i and p_j cannot be discerned from each other by means of the attribute `Colors` in this example information system.
□

Below, we state this more formally.

4.2.2 Indiscernibility of Elements of the Universe

Now that we have defined an equivalence relation on the range of the attributes of $\mathcal{AT}$ we formally show how to induce an equivalence relation on the elements of the universe $\mathcal{U}$.

Two elements $x, y \in \mathcal{U}$ may have equivalent attribute values $a(x)$ and $a(y)$, i. e. $a(x) \approx_a a(y)$, in which case x and y are indiscernible from each other by means of the attribute a. This situation will be denoted $x\theta_a y$. Formally, $x\theta_a y$ is defined as:

$$x\theta_a y \text{ if } \big(\underline{\mathcal{A}_{\sim_a}}(a(x)) = \underline{\mathcal{A}_{\sim_a}}(a(y))\big) \bigwedge \big(\overline{\mathcal{A}_{\sim_a}}(a(x)) = \overline{\mathcal{A}_{\sim_a}}(a(y))\big),$$

or, equivalently,

$$x\theta_a y \text{ if } a(x) \approx_a a(y)$$

In case each attribute a in $\mathcal{AT}$ is considered in its approximation space $(\mathcal{V}_a, \sim_a)$, the equivalence relation $\approx_a$ extends straightforwardly to the *composed equivalence relation* $\approx_A$. This is done by combining $\approx_a$ and by using the projection operation π. Let $\langle U\rangle$ and $\langle V\rangle$ be two tuples of $\prod_{a\in A} 2^{\mathcal{V}_a}$, then the equivalence relation $\approx_A$ is defined as:

$$\langle U\rangle \approx_A \langle V\rangle \text{ if } \bigwedge_{a\in A} \big(\pi_a(\langle U\rangle) \approx_a \pi_a(\langle V\rangle)\big)$$

In this case, the equivalence class of $\langle V\rangle$ in $\approx_A$ is a subset of $\prod_{a\in A} 2^{\mathcal{V}_a}$ and is denoted $[\langle V\rangle]_{\approx_A}$.

Furthermore, two elements of $\mathcal{U}$ may have equivalent A-descriptions in which case they are not discernible from each other by means of their A-descriptions.

Let $A \subseteq \mathcal{AT}$ and $x, y \in \mathcal{U}$, we define an equivalence relation θ_A on the elements $x, y \in \mathcal{U}$ as:

$$x\theta_A y \text{ if } \forall a \in A, \big(\underline{\mathcal{A}_{\sim_a}}(a(x)) = \underline{\mathcal{A}_{\sim_a}}(a(y))\big) \bigwedge \big(\overline{\mathcal{A}_{\sim_a}}(a(x)) = \overline{\mathcal{A}_{\sim_a}}(a(y))\big)$$

Obviously, θ_A is an equivalence relation.

Again, θ_A can be rewritten using $\approx_A$ as:

$$x\theta_A y \text{ if } A(x) \approx_A A(y)$$

Remark

It is worth noting that there is another way to represent an attribute value or description $A(x)$ in our information system. As we argued in Section 4.1, a multi-valued information system implies a relation $R_a \subseteq \mathcal{U} \times \mathcal{V}_a$ where $x\, R_a\, v \Leftrightarrow v \in a(x)$ which can easily be generalized to hold for tuples $A(x)$. Hence, a description $A(x)$ can also be represented as a set of tuples $A(x) \subseteq \prod_{a\in A} \mathcal{V}_a$ instead of a tuple of sets, i. e. $A(x) \in \prod_{a\in A} 2^{\mathcal{V}_a}$ as presented here.

Hence, we may look at some other ways to construct θ_A. For instance, we may define θ_A using equivalence of elements of attribute values. As in [YWL97], we define a composite equivalence relation $\sim_A$ by combining the elementary equivalence relations $\sim_a$. Let $A = \{a_1, \ldots, a_n\}$ and the tuples $\langle u_1, \ldots, u_n\rangle$ and $\langle v_1, \ldots, v_n\rangle$ elements of $\mathcal{V}_1 \times \ldots \times \mathcal{V}_n$. The equivalence relation $\sim_A$ is defined as:

$$\langle u_1, \ldots, u_n\rangle \sim_A \langle v_1, \ldots, v_n\rangle \text{ if } \bigwedge_{a_i \in A} (u_i \sim_{a_i} v_i)$$

This way, θ_A may equivalently be defined as:

$$x\theta_A y \text{ if } \left(\underline{\mathcal{A}_{\sim_A}}(A(x)) = \underline{\mathcal{A}_{\sim_A}}(A(y))\right) \bigwedge \left(\overline{\mathcal{A}_{\sim_A}}(A(x)) = \overline{\mathcal{A}_{\sim_A}}(A(y))\right)$$

Or, we may use Pawlak's rough relation model described in [Paw96]. In this model, given two sets $\mathcal{V}_a$ and $\mathcal{V}_b$, two indiscernibility relations $\sim_a \subseteq \mathcal{V}_a \times \mathcal{V}_a$ and $\sim_b \subseteq \mathcal{V}_b \times \mathcal{V}_b$, we define a new indiscernibility relation by constructing the product of the indiscernibility relations $\sim_a \times \sim_b$.

Now, let us describe some elements of our information system.

Elements of the equivalence classes $[x]_{\theta_A}$ are those elements of the universe of our information system that cannot be discerned from each other by means of their A-descriptions. Often though, we are interested in the description of elements by means of all attributes of the information system. Thus, considering the whole set of attributes $\mathcal{AT}$, the equivalence classes $[x]_{\theta_{\mathcal{AT}}}$ are those subsets of the universe that cannot be discerned in the information system $(\mathcal{U}, \mathcal{AT})$.

Subsequently, to simplify the notation, we will use θ to denote $\theta_{\mathcal{AT}}$ when $\mathcal{AT}$ is understood.

Moreover, throughout this thesis, for convenience, we assume that the equivalence classes of $\mathcal{U}/\theta$ have at least two elements.

4.2.3 Description Language

In the context of an information system we have defined an approximation space in which subsets of the universe may be approximated using lower and upper approximation operators. Here, we define a language that we will use to query elements of the universe by means of their descriptions.

To the information system $(\mathcal{U}, \mathcal{AT})$ we associate a language $\mathcal{L}_{\mathcal{AT}}$. The language is an extension of the one presented in [MT99]. The main difference lies in that in the language we define here we enable the use of approximate descriptions of the elements of the universe in the query.

Our language consists of terms built by means of functor symbols $+$ (sum), $\cdot$ (product) and $-$ (negation). The terms of $\mathcal{L}_{\mathcal{AT}}$ are defined recursively as follows:

- For every attribute $a \in \mathcal{AT}$ and every attribute value $V \subseteq \mathcal{V}_a$, the expression $a = V$ is a term.

- If s and t are terms then so are $s + t$ (sum), $s \cdot t$ (product) and $-s$ (negation).

Terms of $\mathcal{L}_{\mathcal{AT}}$ are used to query the information systems $(\mathcal{U}, \mathcal{AT})$. To allow queries, we define the *value* of a term $a = V$, denoted $|a = V|$, as:

$$|a = V| = \Big\{x \in \mathcal{U} \big| \big(\underline{\mathcal{A}_{\sim_a}}(a(x)) = \underline{\mathcal{A}_{\sim_a}}(V)\big) \bigwedge \big(\overline{\mathcal{A}_{\sim_a}}(a(x)) = \overline{\mathcal{A}_{\sim_a}}(V)\big)\Big\}$$

The value of a query (or term) t to the information system, i. e. $|t|$, is said to be the answer, or result, of a query t. The query value $|a = V|$, or just query for simplicity, returns elements x of the universe whose attribute values $a(x)$ are approximately V, i. e. $a(x) \approx_a V$. We refer to this type of query as *approximate query.*

The value of a sum, product and negation of terms are interpreted as union, intersection and complement, respectively. Hence, $|s + t| = |s| \cup |t|$, $|s \cdot t| = |s| \cap |t|$ and $|-t| = \mathcal{U} \setminus |t|$.

Moreover, query may apply to several attributes at a time: $|(a = V) \cdot (b = V')|$, where $V \subseteq \mathcal{V}_a$ and $V' \subseteq \mathcal{V}_b$, for instance. Such a query can equivalently be written as:

$$|a = V| \cap |b = V'| = \{x | (a(x) \approx_a V) \wedge (b(x) \approx_b V')\} = |A = \langle V, V' \rangle|$$

where $A = \{a, b\}$. To simplify the notation we may use V to denote $\langle V \rangle$ when there is no confusion. We may then define generalized queries of the form $|A = V|$, where V is a tuple of $\prod_{a \in A} 2^{\mathcal{V}_a}$, as:

$$|A = V| = \Big\{x \in \mathcal{U} \big| \big(\underline{\mathcal{A}_{\sim_A}}(a(x)) = \underline{\mathcal{A}_{\sim_A}}(V)\big) \bigwedge \big(\overline{\mathcal{A}_{\sim_A}}(a(x)) = \overline{\mathcal{A}_{\sim_A}}(V)\big)\Big\}$$

Observe that for any $A \subseteq \mathcal{AT}$ and $V \in \prod_{a \in A} 2^{\mathcal{V}_a}$, $|A = V|$ is an element of the quotient set $\mathcal{U}/\theta_A$. Hence, by combining queries we should be able to approximate subsets of the universe $\mathcal{U}$ in the approximation space $(\mathcal{U}, \theta_A)$.

4.2.4 Queries and Approximations

Given an information system $(\mathcal{U}, \mathcal{AT})$ and an approximation space θ_A, subsets $X \subseteq \mathcal{U}$ are approximated by means of queries $|A = V|$.

Let us consider queries with only one attribute a.

To each equivalence class $[x]_{\theta_a} \in \mathcal{U}/\theta_a$ corresponds an equivalence class $[V]_{\approx_a}$ in the quotient set $2^{\mathcal{V}_a}/\approx_a$. We then define two operators that map a subset $X \subseteq \mathcal{U}$ to its lower and upper descriptions, also called *approximate* description.

Let R be a rough set in $\mathcal{R}_{\sim_a}(\mathcal{V}_a)$, we define the *lower description operator* $\underline{\mathcal{D}_{\approx_a}}$ as:

$$\underline{\mathcal{D}_{\approx_a}}(X) = \Big\{R \Big| |a = V| \subseteq X \text{ and } V \in R\Big\}$$

and the *upper description operator* $\overline{\mathcal{D}_{\approx_a}}$ as:

$$\overline{\mathcal{D}_{\approx_a}}(X) = \Big\{R \Big| |a = V| \cap X \neq \emptyset \text{ and } V \in R\Big\}$$

Note that $\underline{\mathcal{D}_{\approx_a}}(X) \subseteq \overline{\mathcal{D}_{\approx_a}}(X)$. The pair $(\underline{\mathcal{D}_{\approx_a}}(X), \overline{\mathcal{D}_{\approx_a}}(X))$ is referred to as the *approximate description* of the set X by means of the attribute a. Also note that, for convenience, the lower and upper description operators return a set of rough sets. Alternatively, instead of a rough set, we would have to return, for each rough set, all its class representatives.

We say that X cannot be described exactly in terms of a-descriptions, or by means of the attribute a, in the information system if $\underline{\mathcal{D}_{\approx_a}}(X) \subsetneqq \overline{\mathcal{D}_{\approx_a}}(X)$.

If an approximate description $(\underline{\mathcal{D}_{\approx_a}}(X), \overline{\mathcal{D}_{\approx_a}}(X))$ of X is available then the set X can be approximated as follows:

$$\bigcup_{V \in \underline{\mathcal{D}_{\approx_a}}(X)} |a = V| \subseteq X \subseteq \bigcup_{V \in \overline{\mathcal{D}_{\approx_a}}(X)} |a = V| \tag{4.1}$$

By $V \in \underline{\mathcal{D}_{\approx_a}}(X)$ (resp. $\overline{\mathcal{D}_{\approx_a}}(X)$) we mean that there exists a rough set R which belongs to $\underline{\mathcal{D}_{\approx_a}}(X)$ (resp. $\overline{\mathcal{D}_{\approx_a}}(X)$) such that $V \in R$.

On the other hand, it may be that the set to be approximated is not known but a pair of approximate descriptions $K, K' \subseteq 2^{\mathcal{V}_a}$ of the set in question is *known.* In this case, an unknown set $X \subseteq \mathcal{U}$ may be approximated as follows:

$$\bigcup_{V \in K} |a = V| \subseteq X \subseteq \bigcup_{V \in K'} |a = V| \tag{4.2}$$

where $K \subseteq K'$. The pair (K, K') is referred to as a *piece of knowledge* about the set X.

In the case above, (K, K') play the same role as the pair of sets of rough sets $(\underline{\mathcal{D}_{\approx_a}}(X), \overline{\mathcal{D}_{\approx_a}}(X))$ that appear in the approximation statement 4.1. Hence, K and K' should be interpreted as sets of class representatives of some rough sets.

To simplify the notation we introduce generalized queries of the form $a = K$ such that:

$$|a = K| = \bigcup_{V \in K} |a = V|$$

Consequently, for any $K \subseteq K'$, the pair $(|a = K|, |a = K'|)$ is a rough set in the approximation space $(\mathcal{U}, \theta_a)$. This can easily be seen since each $|a = V|$ returns, by its construction, an equivalence class of the quotient set $\mathcal{U}/\theta_A$.

We generalize the results above to hold for tuple based queries.

We consider an approximation space θ_A where subsets $X \subseteq \mathcal{U}$ are approximated by means of queries $|A = V|$. The tuples V belong to $\prod_{a \in A} 2^{\mathcal{V}_a}$. This time, to each equivalence class $[x]_{\theta_A}$ corresponds an equivalence class $[V]_{\approx_A}$ in the quotient set $\prod_{a \in A} 2^{\mathcal{V}_a} / \approx_A$. Below, we extend the lower and upper description operators to hold for tuple based queries.

Let R be a rough set in $\prod_{a \in A} \mathcal{R}_{\sim_a}(\mathcal{V}_a)$, the product of the set of all rough sets of the ranges $\mathcal{V}_a$ of each attribute a. We define the *lower description operator* $\underline{\mathcal{D}_{\approx_A}}$ as:

$$\underline{\mathcal{D}_{\approx_A}}(X) = \left\{ R \,\middle|\, |A = V| \subseteq X \text{ and } V \in R \right\}$$

and the *upper description operator* $\overline{\mathcal{D}_{\approx_A}}$ as:

$$\overline{\mathcal{D}_{\approx_A}}(X) = \left\{ R \,\middle|\, |A = V| \cap X \neq \emptyset \text{ and } V \in R \right\}$$

The pair $(\underline{\mathcal{D}_{\approx_A}}(X), \overline{\mathcal{D}_{\approx_A}}(X))$ is referred to as the *approximate description* of the set X by means of the attributes of A. Here too, we have that the pair of approximate descriptions of the set X is such that $\underline{\mathcal{D}_{\approx_A}}(X) \subseteq \overline{\mathcal{D}_{\approx_A}}(X)$.

Let us state the following:

- We say that X cannot be described exactly in terms of A-descriptions in $(\mathcal{U}, \mathcal{AT})$ if:

  $$\underline{\mathcal{D}_{\approx_A}}(X) \subsetneqq \overline{\mathcal{D}_{\approx_A}}(X)$$

- It might be, however, that there exists $B \supsetneqq A$ such that:

  $$\underline{\mathcal{D}_{\approx_B}}(X) = \overline{\mathcal{D}_{\approx_B}}(X)$$

  in this case X can be described exactly in terms of B-descriptions.

- If, for any $A \subseteq \mathcal{AT}$, we have that:

  $$\underline{\mathcal{D}_{\approx_A}}(X) \subsetneqq \overline{\mathcal{D}_{\approx_A}}(X)$$

  then X cannot be described exactly in the information system $(\mathcal{U}, \mathcal{AT})$.

As previously, if an approximate description $(\underline{\mathcal{D}_{\approx_A}}(X), \overline{\mathcal{D}_{\approx_A}}(X))$ of X is available then the set X can be approximated as follows:

$$\bigcup_{V \in \underline{\mathcal{D}_{\approx_A}}(X)} |A = V| \subseteq X \subseteq \bigcup_{V \in \overline{\mathcal{D}_{\approx_A}}(X)} |A = V|$$

If, however, only a description of a set X is known, i. e. a pair (K, K'), where $K, K' \subseteq \prod_{a \in A} 2^{\mathcal{V}_a}$ and $K \subseteq K'$, then the set X may be approximated as follows:

$$\bigcup_{V \in K} |A = V| \subseteq X \subseteq \bigcup_{V \in K'} |A = V|$$

The pair (K, K') is also referred to as a *piece of knowledge* about the set X.

Let us generalize the notation to $A = K$ such that:

$$|A = K| = \bigcup_{V \in K} |A = V|$$

As in the case of single attributes, for any $K \subseteq K'$, the pair $(|A = K|, |A = K'|)$ is a rough set in the approximation space $(\mathcal{U}, \theta_A)$.

Spatial and Attribute Uncertainty

There are two types of uncertainties that can be considered in our information system, *attribute uncertainty* and *spatial uncertainty.*

Attribute uncertainty refers to the uncertainty of an attribute value V in case V is not definable in the approximation space at hand, i. e. $\underline{V} \subsetneqq \overline{V}$.

Spatial uncertainty arises when a subset of the universe $X \subseteq \mathcal{U}$ cannot be described exactly in the language $\mathcal{L}_{\mathcal{AT}}$. This occurs when $\underline{\mathcal{D}_{\approx_A}}(X) \subsetneqq \overline{\mathcal{D}_{\approx_A}}(X)$.

Example 4.2. We build on the example of spatial information system introduced in Example 4.1. The spatial information system $(\mathcal{U}, \mathcal{AT})$ consists of the set of polygons p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8 and p_9, referred to as features, and their attribute $\mathtt{Colors}$. Table 4.1 shows our example spatial information system. The features of the spatial information system are shown in Figure 4.1.

Observe that the example information system presented here is single-granular.

We first look at attribute uncertainty. Consider the polygon p_3, for instance, since we have the following equality:

$$\underline{\mathtt{Color}(p_3)} = \text{gray} = \overline{\mathtt{Color}(p_3)}$$

the attribute value of the feature p_3 is definable. However, for the feature p_5 we have that:

$$\emptyset = \underline{\mathtt{Color}(p_5)} \subsetneqq \overline{\mathtt{Color}(p_5)} = \text{blue}$$

Hence, the attribute value of p_5 is not definable. In other words, no attribute value is known to be part of p_5 for sure, however, 'sky blue' and 'light blue' are *possibly* parts of the attribute values of p_5. In this case we say that the attribute value of p_5 is *uncertain.*

The column '$\mathtt{Approximation}$' of the figure below shows the approximation of the attribute value $\mathtt{Colors}(p_i) = V$ of each feature p_i.

Now let us look at spatial uncertainty. If we consider the features p_1, p_5 and p_7 as a subset of the universe denoted X, showed in light gray in Figure 4.2, then X cannot be described exactly by a query. Indeed, the indiscernible elements of the universe of our example information system are the following four equivalence classes:

$$\mathcal{U}/\theta = \{\, \underbrace{\{p_1, p_3\}}_{1}, \underbrace{\{p_4, p_6, p_8\}}_{2}, \underbrace{\{p_5, p_7\}}_{3}, \underbrace{\{p_2, p_9\}}_{4} \,\}$$

Hence, the set $X = \{p_1, p_5, p_7\}$ is not definable and can only be approximated as shown below:

$$3 \subsetneqq X \subsetneqq 1 \cup 3$$

In this case we say that X is uncertain.

□

Feature	*Colors*	*Approximation*
p_1	‘slate gray’, ‘dark gray’	$\underline{V} =$ gray $= \overline{V}$
p_2	‘lime green’, ‘yellow green’, ‘sea green’	$\underline{V} =$ green $= \overline{V}$
p_3	‘slate gray’, ‘dark gray’	$\underline{V} =$ gray $= \overline{V}$
p_4	‘white’	$\underline{V} =$ white $= \overline{V}$
p_5	‘sky blue’	$\emptyset = \underline{V} \subsetneq \overline{V} =$ blue
p_6	‘white’	$\underline{V} =$ white $= \overline{V}$
p_7	‘light blue’	$\emptyset = \underline{V} \subsetneq \overline{V} =$ blue
p_8	‘white’	$\underline{V} =$ white $= \overline{V}$
p_9	‘lime green’, ‘yellow green’, ‘sea green’	$\underline{V} =$ green $= \overline{V}$

Table 4.1. Example spatial information system

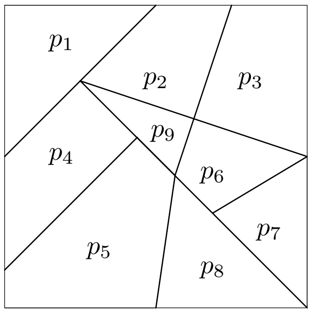


Figure 4.1. Features of the Spatial Information System

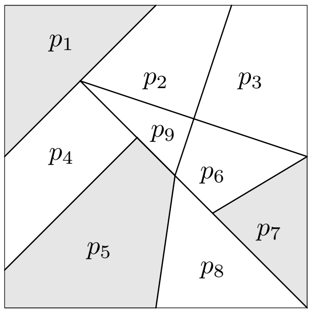


Figure 4.2. Example of Spatial Uncertainty

4.3 Classifications

Here, we introduce a particular case of piece of knowledge called *classification.* A classification has as purpose to group elements of a set into groups or categories according to some knowledge.

Given a single-granular information system $(\mathcal{U}, \mathcal{AT})$ and an attribute a of $\mathcal{AT}$, a classification over $\mathcal{V}_a$, or *attribute classification*, represents a type of knowledge about the information system. This specific type of knowledge about the information system with respect to an attribute $a \in \mathcal{AT}$ is defined to be a set of subsets of $\mathcal{V}_a$, denoted K_I, and defined as:

$$K_I = \left\{ V_i | i \in I,\ \forall i \neq j\ \underline{V_i} \cap \overline{V_j} = \emptyset \right\}$$

The set I is called the *index set* of the classification.

By forming the subsets V_i we are able to express that some attribute values are *reclassified* or assigned to a category i in the index set I.

Let the set K_i be defined as:

$$K_i = \{ E \cap V_i | V_i \in K_I,\ E \in \mathcal{V}_a / \sim_a \}$$

the set K_i is the set of attribute values that are assigned to category i in our single-granular information system.

Since K_i is a set partition of V_i, if an element V_i in the classification K_I is not definable, i. e. $\underline{V_i} \subsetneqq \overline{V_i}$, then some attribute values V of K_i may also turn out not to be definable.

Let us look at interpretations of the elements of K_i.

Let K_I be an attribute classification and, for a fixed i, $V \in K_i$. We use the usual rough set interpretation, hence, given an attribute value $v \in V$ the following interpretation is used:

- if $v \in \underline{V}$ then we say that v certainly belongs to category i, denoted by the classification rule $v \xrightarrow{!} i$.

  This is the case for all attribute values v that belong to V since $V \subseteq [v]$, hence, we extend the interpretation to the attribute value V. That is, $V \xrightarrow{!} i$

- If $v \in \overline{V}$ then we say that v possibly belongs to category i, denoted by the classification rule $v \xrightarrow{?} i$.

  With the same reasoning as above, in this case, we extend the rule as $V \xrightarrow{?} i$.

Note that the condition $\underline{V_i} \cap \overline{V_j} = \emptyset$, used in the definition of K_I, ensures that an attribute value cannot certainly be assigned to a category i and at the same time be possibly assigned to another category j. As a consequence, an attribute value cannot be certainly assigned to two different categories.

Classification rules allow us to represent knowledge without the need to discern all attribute values of an information system. Consider the example below.

Example 4.3. As an example, assume we wish to state that the colors 'lime green', 'yellow green' and 'medium sea green' indicate the presence of vegetation, we build the attribute classification with the following rules:

$$\text{lg} \xrightarrow{!} \text{vegetation},\ \text{yg} \xrightarrow{!} \text{vegetation},\ \text{mg} \xrightarrow{!} \text{vegetation},$$

The knowledge above needs an ability to discern these colors, however, using the generalized rule defined above, using the same settings as of Example 4.1, one can state:

$$\text{green} \xrightarrow{!} \text{vegetation}$$

The 'green' color, in accordance with the equivalence relation that we have chosen, is also relatively easy to discern.

Now suppose we have some additional knowledge about identification of vegetation wetness, wet vegetation and dry vegetation for instance. Assume that the knowledge is set as follows: 'medium sea green' indicates wet vegetation and 'lime green' *may* indicate wet vegetation as well as dry vegetation and 'yellow green' indicates dry vegetation. The attribute classification is built as:

$$\begin{aligned} K_{\text{wetness}} = \Big\{ & V_{\text{wet}} = \{\text{medium sea green}, \text{lime green}\}, \\ & V_{\text{dry}} = \{\text{lime green}, \text{yellow green}\} \Big\} \end{aligned}$$

V_{wet} and V_{dry} are not definable since $\emptyset = \underline{V_{\text{wet}}} = \underline{V_{\text{dry}}} \subsetneq \overline{V_{\text{wet}}} = \overline{V_{\text{dry}}} = \text{green}$. The attribute classification induces the following rules:

$$\begin{aligned} & \text{medium sea green} \xrightarrow{?} \text{wet},\ \text{lime green} \xrightarrow{?} \text{wet}, \\ & \text{lime green} \xrightarrow{?} \text{dry}, \text{yellow green} \xrightarrow{?} \text{dry}, \end{aligned}$$

Again, since the knowledge above needs an ability to discern 'medium sea green', 'lime green' and 'yellow green' from each other, we may use the generalized rules:

$$\text{green} \xrightarrow{?} \text{wet and green} \xrightarrow{?} \text{dry}$$

Hence, the information system together with our knowledge do not provide enough information to discern between wet and dry vegetation by means of the green color. As a consequence, parcels of lands with wet vegetation and parcels of land with dry vegetation become uncertain since they cannot be discerned from each other.
□

Below we show how to use attribute classifications in queries to the information system.

4.4 Querying Classifications

Given an information system $(\mathcal{U}, \mathcal{AT})$ and a classification K_I over $\mathcal{V}_a$, we wish to query the information system about elements of the universe according to the categories of I. In other terms, we would like to retrieve elements of the universe that have been classified to a certain category $i \in I$. For this purpose we define two query operators, the lower and the upper query operators defined respectively as:

$$\lfloor a = V_i \rfloor = \Big\{ |a = E| \;\Big|\; E \subseteq V_i,\ E \in \mathcal{V}_a/\sim_a \Big\}$$
$$\lceil a = V_i \rceil = \Big\{ |a = E \cap V_i| \;\Big|\; E \cap V_i \neq \emptyset,\ E \in \mathcal{V}_a/\sim_a \Big\}$$

Note that $\lfloor a = V_i \rfloor \subseteq \lceil a = V_i \rceil$ and that $(\lfloor a = V_i \rfloor, \lceil a = V_i \rceil)$ is a rough set in the approximation space $(\mathcal{U}, \theta_a)$. We say that it is a rough set built out of queries.

We define a *universal classification* as a classification over the universe $\mathcal{U}$ that is induced by an attribute classification K_I. The universal classification induced by K_I is denoted $[K_I]$ and is defined to be a set of rough sets $(\lfloor a = V_i \rfloor, \lceil a = V_i \rceil)$, where $V_i \in K_I$.

A rough set $(\lfloor a = V_i \rfloor, \lceil a = V_i \rceil)$ is characterized follows:

- the *lower classification query* $\lfloor a = V_i \rfloor$ is said to be the lower approximation of the rough set and returns elements of the universe whose attribute values surely belong to i.
- the *upper classification query* $\lceil a = V_i \rceil$ is said to be the upper approximation of the rough set and returns elements of the universe whose attribute values possibly belong to i.

It is important to note that, in general, as for usual rough sets, the following inequalities hold for rough sets built out of queries:

$$\lfloor a = V_i \cup V_j \rfloor \neq \lfloor a = V_i \rfloor \cup \lfloor a = V_j \rfloor$$
$$\lceil a = V_i \cap V_j \rceil \neq \lceil a = V_i \rceil \cap \lceil a = V_j \rceil$$

Moreover, upper and lower classification queries approximate an unknown set $X \subseteq \mathcal{U}$, hence, we do not have a representative for the rough set $R = (\lfloor a = V_i \rfloor, \lceil a = V_i \rceil)$. It turns out, however, that in the case of upper and lower classification query operations used with single-granular attribute values, i. e. $V_i/\sim_a$ is a singleton for all i, the attribute value V_i may be considered as a set representative for R. As a consequence, union and intersection of lower and upper queries may be truth-functional if we restrict the domain to $(\mathcal{V}_a)_{AB}$ using the truth-functional set representatives of Section 3.2.3, for instance. If all the elements V_i of an attribute classification K_I belong to $(\mathcal{V}_a)_{AB}$, then, the following hold:

$$\lfloor a = V_i \cup V_j \rfloor = \lfloor a = V_i \rfloor \cup \lfloor a = V_j \rfloor \tag{4.3}$$
$$\lceil a = V_i \cap V_j \rceil = \lceil a = V_i \rceil \cap \lceil a = V_j \rceil \tag{4.4}$$

The attribute classification is said to be truth-functional.

Thus, rough sets built out of queries may be combined component wise as:

$$\begin{aligned}&(\lfloor a = V_i \rfloor, \lceil a = V_i \rceil) \cap (\lfloor a = V_j \rfloor, \lceil a = V_j \rceil)\\ &\qquad = (\lfloor a = V_i \rfloor \cap \lfloor a = V_j \rfloor, \lceil a = V_i \rceil \cap \lceil a = V_j \rceil)\\ &(\lfloor a = V_i \rfloor, \lceil a = V_i \rceil) \cup (\lfloor a = V_j \rfloor, \lceil a = V_j \rceil)\\ &\qquad = (\lfloor a = V_i \rfloor \cup \lfloor a = V_j \rfloor, \lceil a = V_i \rceil \cup \lceil a = V_j \rceil)\end{aligned}$$

where $\forall i \, V_i \in (\mathcal{V}_a)_{AB}$.

Attribute and universal classifications extend to descriptions $A \subseteq \mathcal{AT}$ where the attribute values are tuples V_i of $\prod_{a \in A} 2^{\mathcal{V}_a}$. When tuples are considered there is a notion of *coarseness* in the queries defined above. For instance, assume a classification K_I over $\prod_{a \in A} 2^{\mathcal{V}_a}$, then, given $B \subseteq A$, the following properties hold:

$$\lfloor A = V_i \rfloor \subseteq \lfloor B = \pi_B(V_i) \rfloor \tag{4.5}$$

$$\lceil A = V_i \rceil \subseteq \lceil B = \pi_B(V_i) \rceil \tag{4.6}$$

Recall that π_B is the generalized projection operator defined in Section 4.1.

In the situation above, the query $\lfloor A = V_i \rfloor$ (respectively $\lceil A = V_i \rceil$) is said to be *finer* than $\lfloor B = \pi_B(V_i) \rfloor$ (respectively $\lceil B = \pi_B(V_i) \rceil$). Conversely, the query $\lfloor B = \pi_B(V_i) \rfloor$ (respectively $\lceil B = \pi_B(V_i) \rceil$) is said to be *coarser* than $\lfloor A = V_i \rfloor$ (respectively $\lceil A = V_i \rceil$).

Given the universal classification $[K_I] = \{(\lfloor \mathcal{AT} = V_i | \rfloor, \lceil \mathcal{AT} = V_i \rceil) | V_i \in K_I\}$, if $\lceil \mathcal{AT} = V_i \rceil \cap \lceil \mathcal{AT} = V_j \rceil \neq \emptyset$, then the attribute classification K_I does not provide enough information to discern between the elements of class i and the elements of class j.

Let $A \subseteq \mathcal{AT}$, the universal classification $[K_I]_A$ over $\mathcal{U}$ is said to be *crisp* if:

$$\forall i \in I, \lfloor A = V_i \rfloor = \lceil A = V_i \rceil$$

otherwise, $[K_I]_A$ is said to be a *rough* classification. For a crisp classification $[K_I]_A$, the element $\lfloor A = V_i \rfloor = \lceil A = V_i \rceil$ is referred to as elements of class i.

Example 4.4. We continue on Example 4.2. Recall that we have the following attribute values in our information system:

$$\mathcal{V}_c / \sim = \{ \underbrace{\{\text{lg}, \text{yg}, \text{mg}\}}_{\text{green}}, \underbrace{\{\text{sb}, \text{lb}\}}_{\text{blue}}, \underbrace{\{\text{sg}, \text{dg}\}}_{\text{gray}}, \{\text{white}\} \}$$

Suppose that according to some knowledge, colors 'lime green', 'yellow green', 'medium sea green' and 'sky blue' indicate the presence of vegetation, whereas colors 'light blue', 'slate gray', 'dark gray' indicate the absence of vegetation. Using this knowledge, one can build an attribute classification K_I as follows:

$$K_I = \{ \underbrace{\{\text{lg}, \text{yg}, \text{mg}, \text{sb}\}}_{V_1}, \underbrace{\{\text{lb}, \text{sg}, \text{dg}\}}_{V_2} \}$$

Here, $I = \{1, 2\}$ is the index set of the classification, where V_1 is the set of attribute values that indicates presence of vegetation and V_2 indicating absence of vegetation.

The rules of the attribute classification are as follows:

$$\text{green} \xrightarrow{!} \text{Veg}$$
$$\text{gray} \xrightarrow{!} \text{No Veg}$$
$$\text{blue} \xrightarrow{?} \text{Veg}$$
$$\text{blue} \xrightarrow{?} \text{No Veg}$$

According to the rules above, the green color certainly indicates the presence of vegetation whereas the blue color may indicate both the presence and absence of vegetation. The gray color, on the other hand, indicates the absence of vegetation.

Hence, the universal classification is built up as:

$$[K_I] = \{(\lfloor a = V_1 \rfloor, \lceil a = V_1 \rceil), (\lfloor a = V_2 \rfloor, \lceil a = V_2 \rceil)\}$$

where the queries return the following elements of the universe:

$$\lfloor a = V_1 \rfloor = \{p_2, p_9\}$$
$$\lceil a = V_1 \rceil = \{p_2, p_9, p_5, p_7\}$$
$$\lfloor a = V_2 \rfloor = \{p_1, p_3\}$$
$$\lceil a = V_2 \rceil = \{p_1, p_3, p_5, p_7\}$$

Figure 4.3 shows the example universal classification. Figure 4.3(a) shows in dark gray the features that surely classify to area with vegetation, and in light gray the area of uncertainty, that is those areas that possibly have vegetation. Figure 4.3(b) shows in dark gray the features that surely classify to area with no vegetation and in light gray the area of uncertainty, that is those areas that possibly do not have vegetation.
□

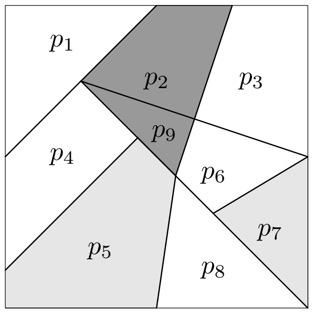
4.5 Rough Fuzzy Hybridization

It is possible to integrate fuzzy information into attribute classifications by using rough fuzzy sets, see Section 3.4. Given an information system $(\mathcal{U}, \mathcal{AT})$ the knowledge about the information system may be expressed by an attribute classification K_I where the sets V_i are fuzzy sets. The sets V_i may be tuples $V_i \subseteq \prod_{a \in A} 2^{\mathcal{V}_a}$.

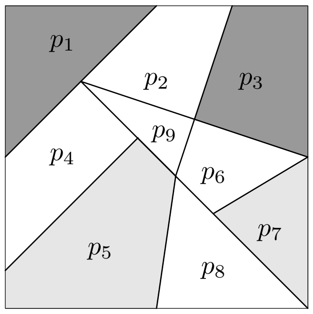
$$K_I = \{\mu_{V_i} \mid i \in I\}$$

where $\mu_{V_i} : \mathcal{V}_a \to [0, 1]$ is a membership function that determines the degree of membership of an elements v of $\mathcal{V}_a$ in the set V_i. The set K_I is referred to as *rough fuzzy attribute classification*.

We use the previous semantic for attribute classifications but this time we have an additional information about the degree of membership of the attribute values.



(a) Polygons with Vegetation.



(b) Polygons with No Vegetation.

Figure 4.3. Example Universal Classification

Hence, subsets V_i express that some attribute values, those discernible in V_i, are assigned, or reclassified, to a category $i \in I$ with a degree of membership μ_{V_i} according to some knowledge. By using fuzzy sets additional information about the degree of membership of some fuzzy concept can be expressed in rough fuzzy attribute classifications.

For semantic reasons, as for rough attribute classifications, we impose some additional conditions on rough fuzzy attribute classifications, that is:

$$\forall i \neq j, v \in \mathcal{V}_a, \text{ if } \mu_{\underline{V_i}}(v) = 1 \text{ then } \mu_{\overline{V_j}}(v) = 0$$

A classification over the universe $\mathcal{U}$ may be induced by a rough fuzzy attribute classification K_I. To achieve this we use the α-level sets of the sets V_i, i. e. $(V_i)_\alpha$, in lower and upper classification query operations in the same way as for rough attribute classifications. For each $0 \leq \alpha \leq 1$, we get a universal classification

$[(K_I)_\alpha]$ over $\mathcal{U}$ defined as:

$$\big[(K_I)_\alpha\big] = \Big\{\big(\lfloor a = (V_i)_\alpha \rfloor, \lceil a = (V_i)_\alpha \rceil\big) \;\big|\; \alpha \in [0,1],\ i \in I\Big\}$$

4.6 Conclusion

We have defined a query language based on the one by Marek et al. [MT99]. Our query language is more powerful in the sense that it allows the representation of attribute uncertainty. Moreover, in our model, there is a clear relation between attribute and spatial uncertainty.

Based on queries of attribute classifications, we have defined rough and rough fuzzy universal classifications. We have shown that they are suitable to represent uncertainty in classifications.

We have shown that intersection and union of queries can be made truth-functional and that, as a consequence, rough sets built out of queries can be combined through intersection and union operators. Truth-functionality turns out to be an important property as it is necessary to intersect elements of a universal classification to measure the areas of uncertainty. When intersection is performed over all the elements of a universal classification we say that we perform an *overlay* operation. Chapter 7 presents some experiments on the use of the overlay operation.

Chapter 5

Accuracy Assessments

In this chapter we present ways to assess the accuracy of classifications. Section 5.1 recalls some methods used to assess accuracy of crisp classifications. Sections 5.2 and 5.3 present some methods to assess accuracy of rough, respectively rough fuzzy, classifications. Finally, in Section 5.4, we present a way to assess topological relations between rough classifications.

5.1 Accuracy Assessment of Crisp Classifications

Let $(\mathcal{U}, \mathcal{AT})$ be an information system, and let X_I and Y_I be two attribute classifications over $\prod_{a \in A} 2^{\mathcal{V}_a}$, i. e. $X_i, Y_i \subseteq \prod_{a \in A} 2^{\mathcal{V}_a}$, inducing two crisp classifications $[X_I]$ and $[Y_I]$ over $\mathcal{U}$. Recall that for a crisp classification lower and upper approximations are equal, i. e. $\lfloor X_i \rfloor = \lceil X_i \rceil$, denoted X_i. Suppose we wish to compare classification $[Y_I]$ against $[X_I]$, i. e. see how well X_i and Y_i match for all $i \in I$. The reference classification is chosen to be $[X_I]$ and $[Y_I]$ is the classification that is being assessed. To achieve the comparison we will be using an error matrix as described in [Con91].

First, we define the cardinality, or total area, of a crisp classification $[X_I]$ as:

$$\big|[X_I]\big| = \big|\bigcup_{i \in I} X_i\big| = \sum_{i \in I} \big|X_i\big|.$$

The error matrix is defined to be a $|I| \times |I|$ matrix where the columns correspond to the *reference* classification $[X_I]$ and the rows to the classification that is to be assessed, i. e. $[Y_I]$. The cells of the matrix are elements $z_{i,j}$ such that $z_{i,j} = |X_i \cap Y_j|$. The error matrix has the following three properties:

- The *row-sum* property: $\sum_j z_{i,j} = |Y_i|$, i. e. the sum of the elements in a single row of the matrix is the area of the corresponding element of $[Y_I]$.

- The *column-sum* property: $\sum_i z_{i,j} = |X_j|$, i. e. the sum of the elements in a single column of the matrix is the area of the corresponding element of $[X_I]$.
- The *total-sum* property: $\sum_{i,j} z_{i,j} = \big|[X_I]\big| = \big|[Y_I]\big|$, i. e. the sum of all the elements of the matrix is the area covered by either of the classifications $[X_I]$ or $[Y_I]$.

Let $A = \big|[X_I]\big| = \big|[Y_I]\big|$ be the total area of the classifications. The *overall accuracy* is defined as the total match between the two classifications divided by the total area of the classifications:

$$A_o = \sum_i z_{i,i}/A \tag{5.1}$$

In a similar manner, one can compute the accuracy of individual classes with the choice of dividing the correctly classified area, i. e. $z_{i,i}$ for class i, by either $|X_i|$ or $|Y_i|$. Assuming that the reference classification is associated with the columns of the matrix, dividing by the column total gives the *omission error*, also called *producer's accuracy*; dividing by the row total gives the *commission error*, also called *user's accuracy*. They are defined respectively as follows:

$$O_j = z_{j,j}/\sum_i z_{i,j}, \text{ and } C_i = z_{i,i}/\sum_j z_{i,j} \tag{5.2}$$

5.2 Accuracy Assessment of Rough Classifications

We present some measures that can be used to assess accuracy of rough classifications.

5.2.1 The Overlap and the Crispness Measures

The *overlap measure*, denoted $\mathcal{O}$, is a measure that applies to a single rough classification $[X_I]$ measuring the degree of overlap. $\mathcal{O}$ is defined as:

$$\mathcal{O} = \big(\sum_{i \in I} |\lceil X_i \rceil| - A\big)/A$$

where A is the total area of the classification $[X_I]$; $|I|$ is the number of classes or attributes of the classification. The degree of overlap increases with the amount of overlapping upper approximations $\lceil X_i \rceil$ indicating the degree of uncertainty. The more there is overlap, the more uncertain the classification is.

The interval $0 \leq \mathcal{O} \leq |I| - 1$ is interpreted as follows: if the overlap measure is equal to $|I| - 1$ the rough classification is completely rough, i. e. the lower approximations $\lfloor X_i \rfloor$ are all empty. An overlap measure equal to 0, on the other hand, indicates that the rough classification is completely crisp, i. e. areas of uncertainty $(\lceil X_i \rceil \setminus \lfloor X_i \rfloor)$ are all empty.

A similar measure is the *crispness measure* $\mathcal{C}$ defined as:

$$\mathcal{C} = \big(\sum_{i \in I} |\lfloor X_i \rfloor|\big)/A$$

It takes values between 0, which indicates that the classification is completely uncertain, and 1, which indicates that the classification is completely crisp.

5.2.2 Comparing Rough Classifications

As previously, we have two rough classifications over $\mathcal{U}$, $[X_I]$ and $[Y_I]$, and we wish to compare $[Y_I]$ against the reference classification $[X_I]$. In a rough classification $[X_I]$, we might have that $\lfloor X_i \rfloor \neq \lceil X_i \rceil$ for some attribute $i \in I$, hence, we cannot directly apply the accuracy measures seen in Section 5.1. The reason is that $\big|[X_I]\big| = \big|\bigcup_{i \in I} \lceil X_i \rceil\big| \neq \sum_{i \in I} \big|\lceil X_i \rceil\big|$. Indeed, in a rough classification $[X_I]$, the upper approximation of two different classes $\lceil X_i \rceil$ and $\lceil X_j \rceil$ may overlap, i. e. $\lceil X_i \rceil \cap \lceil X_j \rceil \neq \emptyset$. Hence, some element might be counted more than once when calculating $\big|[X_I]\big|$. To overcome this we transform rough classifications into virtual crisp ones, also called parameterized crisp classifications.

5.2.3 Parameterized Crisp Classifications

To transform a rough classification into a parameterized crisp classification we introduce a parameter on subsets that are shared between some attributes $C \subseteq I$, we call them rough components. A *rough component* is a subset of $\mathcal{U}$ that is in all the areas of uncertainty of the classes that are members of C, and in none of the areas of uncertainty of any of the classes that are not members of C. Hence, a rough component is defined as:

$$R_C = \bigcap_{i \in C} \big(\lceil X_i \rceil \setminus \lfloor X_i \rfloor\big) \setminus \bigcup_{i \notin C} \big(\lceil X_i \rceil \setminus \lfloor X_i \rfloor\big).$$

To each class, or attribute $i \in I$, in a rough classification $[X_I]$, corresponds a set of rough components Γ_i, whose elements are referred to as rough components of class i, defined as:

$$\Gamma_i = \{R_C | i \in C, C \subseteq I\}$$

Hence, by considering the possible assignments of the elements of Γ_i to the shared classes we can arrive at a crisp classification that is parameterized over the assignment possibilities. The parameterization is achieved by using a set of $|I|$ parameter functions $f_i, i \in I$, associated with the classification $[X_I]$. Each parameter function f_i corresponds to a class, or attribute $i \in I$. We require the parameter functions f_i to satisfy the following properties:

$$\forall A \subseteq \mathcal{U}, \bigcup_{i \in I} f_i(A) = A$$

$$\forall i, \forall A, B \subseteq \mathcal{U}, f_i(A \cup B) = f_i(A) \cup f_i(B)$$
$$\forall i, \forall A, B \subseteq \mathcal{U}, f_i(A \cap B) = f_i(A) \cap f_i(B)$$
$$\forall i, j, i \neq j, \forall A, B \subseteq \mathcal{U}, f_i(A) \cap f_j(B) = \emptyset$$

Now we are able to express the rough classification $[X_I]$ in terms of a *parameterized crisp classification* X_I as:

$$X_I = \big\{ \lfloor X_i \rfloor \cup f_i(\Gamma_i) | i \in I \big\}.$$

We are now able to compute the cardinality $\big|[X_I]\big|$ since there is no overlap between upper approximations, i. e. $\forall i, j, i \neq j, f_i(\Gamma_i) \cap f_j(\Gamma_j) = \emptyset$. Functions f_i translate to scalar parameters as follows:

$$\forall A \subseteq \mathcal{U}, |A| = |\bigcup_{i \in I} f_i(A)| = \sum_{i \in I} |f_i(A)|,$$

hence, $|f_i(A)| = x_i \cdot |A|$ where $0 \leq x_i \leq 1$ and $\sum_{i \in I} x_i = 1$.

Applied to the cardinality of the parameterized crisp classification X_I, $|X_I|$ is the sum of the cardinalities of all classes X_i, thus, $|X_I| = \sum_{i \in I} |X_i|$. The cardinality of a class X_i depends on the parameter function f_i:

$$\forall i \in I, |X_i| = \big|\lfloor X_i \rfloor \cup f_i(\Gamma_i))\big| = \big|\lfloor X_i \rfloor\big| + |f_i(\Gamma_i)| = \big|\lfloor X_i \rfloor\big| + \sum_{K \in \Gamma_i} x_{i,K} \cdot |K|,$$

we also have to put some restrictions on the parameters $x_{i,K}$:

$$\forall K \in \Gamma_i,\ 0 \leq x_{i,K} \leq 1 \text{ and } \sum_i x_{i,K} = 1$$

5.2.4 Parameterized Error Matrix

It is possible to build an error matrix to compare a rough classification $[X_I]$ with a crisp classification Y_I or to compare two rough classifications $[X_I]$ and $[Y_I]$ with each other. In both cases we build a $|I| \times |I|$ matrix with elements $x_{i,j}$. In the case where $[X_I]$ is rough and $[Y_I]$ is crisp the element $x_{i,j}$ writes as:

$$\begin{aligned}
x_{i,j} &= |X_i \cap Y_j| = \big|\big(\lfloor X_i \rfloor \cup f_i(\Gamma_i)\big) \cap Y_j\big| = \big|\lfloor X_i \rfloor \cap Y_j\big| + |f_i(\Gamma_i) \cap Y_j| \\
&= \big|\lfloor X_i \rfloor \cap Y_j\big| + \Big|\bigcup_{K\in\Gamma_i} \big(f_i(K)\big) \cap Y_j\Big| = \big|\lfloor X_i \rfloor \cap Y_j\big| + \Big|\bigcup_{K\in\Gamma_i} \big(f_i(K) \cap Y_j\big)\Big| \\
&= \big|\lfloor X_i \rfloor \cap Y_j\big| + \Big|\bigcup_{K\in\Gamma_i} \big(f_i(K \cap Y_j)\big)\Big| = \big|\lfloor X_i \rfloor \cap Y_j\big| + \sum_{K\in\Gamma_i} x_{i,K} \cdot |K \cap Y_j|
\end{aligned} \tag{5.3}$$

where $f_i \in \mathcal{F}_{X_I}$ and the following conditions on the parameters hold:

$$0 \leq x_{i,K} \leq 1 \text{ and } \sum_i x_{i,K} = 1.$$

If both $[X_I]$ and $[Y_I]$ are rough, $[Y_I]$ should also be transformed to a parameterized crisp classification. We therefore introduce R'_C defined as:

$$R'_C = \bigcap_{i\in C} \big(\lceil Y_i \rceil \setminus \lfloor Y_i \rfloor\big) \setminus \bigcup_{i\notin C} \big(\lceil Y_i \rceil \setminus \lfloor Y_i \rfloor\big),$$

and the set of rough components of $[Y_I]$ defined as $\Delta_i = \{R'_C | i \in C, C \subseteq I\}$. We then use the parameterized crisp classification $Y_I = \big\{\lfloor Y_i \rfloor \cup g_i(\Delta_i) | i \in I\big\}$, where $g_i, i \in I$, are parameter functions associated with classification $[Y_I]$ with the same properties as the parameter function f_i above. In this situation, $x_{i,j}$ writes as:

$$\begin{aligned}
x_{i,j} &= |X_i \cap Y_j| = \big|\big(\lfloor X_i \rfloor \cup f_i(\Gamma_i)\big) \cap \big(\lfloor Y_j \rfloor \cup g_j(\Delta_j)\big)\big| \\
&= \big|\lfloor X_i \rfloor \cap \lfloor Y_j \rfloor\big| + \sum_{K\in\Gamma_i} x_{i,K} \cdot \big|K \cap \lfloor Y_j \rfloor\big| + \sum_{L\in\Delta_j} x_{j,L} \cdot \big|\lfloor X_i \rfloor \cap L\big| \\
&\quad + \sum_{K\in\Gamma_i} \sum_{L\in\Delta_j} |f_i(K) \cap g_j(L)| \\
&= \big|\lfloor X_i \rfloor \cap \lfloor Y_j \rfloor\big| + \sum_{K\in\Gamma_i} x_{i,K} \cdot \big|K \cap \lfloor Y_j \rfloor\big| + \sum_{L\in\Delta_j} x_{j,L} \cdot \big|\lfloor X_i \rfloor \cap L\big| \\
&\quad + \sum_{K\in\Gamma_i} \sum_{L\in\Delta_j} |f_i(K \cap g_j(L))| \\
&= \big|\lfloor X_i \rfloor \cap \lfloor Y_j \rfloor\big| + \sum_{K\in\Gamma_i} x_{i,K} \cdot \big|K \cap \lfloor Y_j \rfloor\big| + \sum_{L\in\Delta_j} x_{j,L} \cdot +\big|\lfloor X_i \rfloor \cap L\big| \\
&\quad + \sum_{K\in\Gamma_i} \sum_{L\in\Delta_j} x_{i,K} \cdot |K \cap g_j(L))| \\
&= \big|\lfloor X_i \rfloor \cap \lfloor Y_j \rfloor\big| + \sum_{K\in\Gamma_i} x_{i,K} \cdot \big|K \cap \lfloor Y_j \rfloor\big| + \sum_{L\in\Delta_j} x_{j,L} \cdot \big|\lfloor X_i \rfloor \cap L +\big| \\
&\quad + \sum_{K\in\Gamma_i} \sum_{L\in\Delta_j} x_{i,K} \cdot x_{j,L} \cdot |K \cap L|
\end{aligned} \tag{5.4}$$

with the following conditions on the parameters:

$$0 \leq x_{i,K} \leq 1 \text{ and } \sum_i x_{i,K} = 1$$
$$0 \leq x_{j,L} \leq 1 \text{ and } \sum_j x_{j,L} = 1$$

Computing the error matrix using the formulas 5.3 and 5.4 gives a parameterized error matrix with a set of conditions on each parameter. All accuracy measures seen in Section 5.1 can be used on the parameterized error matrix.

Let us look at an example of parameterized error matrix for the case of a rough classification $[X_I]$ compared to a crisp classification $[Y_I]$.

Example 5.1. Figure 5.1 shows an example comparison of a crisp and a rough classification. In the example the crisp classification, which is the reference classification, has two crisp classes A and B. The crisp class A is delimited by a plain line, the crisp class B is delimited by a bold line. Rough classes are delimited by dashed lines, the rough class A by long dashes with a shaded rectangle as lower approximation. Rough class B is delimited by short dashes with a shaded triangle as lower approximation. The number on the figure show area sizes. The resulting parameterized error matrix is given below. Note that the indices of the error matrix below have been simplified somewhat:

	A	B
A	$10 + 11x_1 + 4x_2$	$4x_3 + 4x_4$
B	$5x_6 + 3x_8$	$4 + 10x_5 + 5x_7$

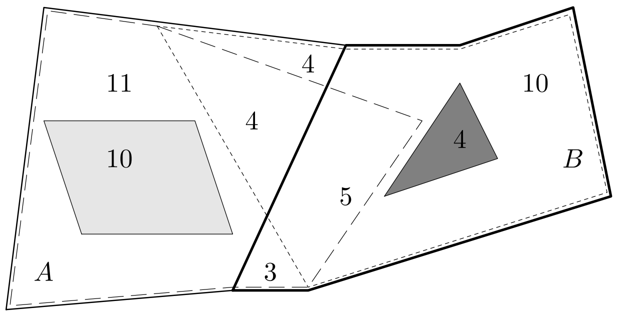


Figure 5.1. Comparing Rough and Crisp Classification

□

5.3 Accuracy Assessment of Rough Fuzzy Classifications

We develop some measures to be applied in the case when fuzziness is used in combination with rough classifications.

5.3.1 Rough Fuzzy Measures

Assume we have a rough attribute classification X_I to which we want to combine some fuzzy concept. This leads to a rough fuzzy classification X_I with fuzzy sets μ_{X_i}. This way, we could, for instance, look at how much fuzziness is contained in the classification X_I.

We first have to find a way to measure how fuzzy a given data-point is. Let $g : [0,1] \to [0,1]$ such that $\forall x \in (0,1), g(x) > 0$, $g(0) = g(1) = 0$ and $g(0.5) = 1$. The function g defines the degree to which a data-point x is fuzzy. If x is close to 0 or 1, $g(x)$ is close to 0 (not fuzzy); if x is close to 0.5, $g(x)$ is close to 1 (fuzzy). For instance, we can choose $g(x) = 4 \cdot x \cdot (1 - x)$.

Let Id be the indicator function $\mathrm{Id} : \mathbb{R} \to \{0,1\}$ defined as $\mathrm{Id}(x) = 0$ if $x = 0$; $\mathrm{Id}(x) = 1$ otherwise. We define the lower degree of fuzziness f of a rough fuzzy classification X_I as:

$$f = \sum_{i \in I} \sum_{v \in X_i} g(x_i) \Big/ \sum_{i \in I} \sum_{v \in X_i} \mathrm{Id}(x_i)$$

where x_i has been defined as $x_i = \mu_{\underline{X_i}}(v)$ to simplify the formula above. The upper degree of fuzziness $\mathcal{F}$ is defined as:

$$\mathcal{F} = \sum_{i \in I} \sum_{v \in X_i} g(\mathcal{X}_i) \Big/ \sum_{i \in I} \sum_{v \in X_i} \mathrm{Id}(\mathcal{X}_i)$$

here too, to simplify the formula above, we pose $\mathcal{X}_i = \mu_{\overline{X_i}}(v)$.

Both f and $\mathcal{F}$ are specifically defined to be 0 whenever $\forall i \in I$, $x_i = 0$, respectively $\mathcal{X}_i = 0$.

Measures f and $\mathcal{F}$ are the ratio of fuzzy areas to the total areas of the classification. Their values vary between 0 and 1, 0 indicates a completely non-fuzzy classification, whereas 1 indicates a completely fuzzy classification.

Another measure of interest is the roughness measure r defined as:

$$r = \sum_{i \in I} \sum_{v \in X_i} (\mathcal{X}_i - x_i) \Big/ \sum_{i \in I} \sum_{v \in X_i} \mathrm{Id}(x_i)$$

where x_i and $\mathcal{X}_i$ are reused with the same definitions as above.

As for the range of the roughness measure r, since the following inequality holds:

$$0 \leq \mu_{\overline{X_i}}(v) - \mu_{\underline{X_i}}(v) \leq \mu_{\overline{X_i}}(v)$$

we can deduce that $0 \leq r \leq 1$. The value 0 indicates a crisp classification, whereas 1 indicates a completely rough classification.

Fuzziness and roughness measures have been first introduced in our joint paper [AKO03]. Here, we presented a modified version to fit our concept of information system.

5.3.2 Parameterized Error Matrix for Rough Fuzzy Classifications

If we wish to compare two rough fuzzy classifications with each other, say $[X_I]$ and $[Y_I]$ with $[X_I]$ being the reference classification as previously, it is possible to build a parameterized rough fuzzy error matrix in a straightforward way. The standard first step is to consider the α-level sets $\lfloor (X_i)_\alpha \rfloor$ and $\lceil (X_i)_\alpha \rceil$ of the classification $[X_I]$. We then proceed similarly for the classification $[Y_I]$ for the same value of α. This way we arrive at the same parameterized error matrix as for the rough classification case. Hence the elements, or cells, $x_{i,j}$ of the matrix are retrieved using the formulas 5.3 and 5.4.

We should however remember that we proceeded with a specific value of α, thus, it might be necessary to pick several values for α and compare the classifications $[X_I]$ and $[Y_I]$ to each other for all of those values.

5.4 Topological Characterization

Sometimes we do not wish to assess accuracies numerically, rather, we want to know for instance if two sets $X_i, Y_i \subseteq \prod_{a\in A} 2^{\mathcal{V}_a}$ of two attribute classifications X_I and Y_I lead to overlapping sets when used in a query, i. e. $\lceil X_i \rceil \cap \lceil Y_i \rceil \neq \emptyset$. In this case we could assert that X_i and Y_i overlap, denoted $(X_i, \mathit{overlap}, Y_i)$. The assertion $(X_i, \mathit{overlap}, Y_i)$ is referred to as a topological relation [CF96, CF97].

Note that it is the result of the queries $\lceil X_i \rceil$ and $\lceil Y_i \rceil$ which overlap. However, we will always refer to the descriptions used in the query when we assert topological relations.

Clementini et al. in [CFvO93] introduced the Calculus-Based Method, a small set of formal topological relations. The Calculus-Based Method is used to assert topological relations between point-sets of $\mathbb{R}^2$ through the topological relations *meet*, *in*, *overlap* and *disjoint*. We extend the Calculus-Based Method for rough sets $(\lfloor X \rfloor, \lceil X \rceil)$, where $X \subseteq \prod_{a\in A} 2^{\mathcal{V}_a}$.

We will use the three-valued Kleene logic [Kle57] to combine assertions through conjunctions, disjunctions and negations.

We first define our three-valued logic-based language. The language will consist of a set of propositions $\mathcal{P}$ of the form $X = \emptyset$, where $X \subseteq \prod_{a\in A} 2^{\mathcal{V}_a}$, two binary connectives $\wedge$ and $\vee$, and a unary connective $\neg$. We define a *valuation function* $v : \mathcal{P} \to \{\mathrm{T}, \mathrm{F}, \mathrm{M}\}$, where T, F, M represent 'True', 'False' and 'Maybe' respectively as follows:

$$v(X = \emptyset) = \begin{cases} \text{T} & \text{if } \lceil X \rceil = \emptyset \\ \text{F} & \text{if } \lfloor X \rfloor \neq \emptyset \\ \text{M} & \text{otherwise} \end{cases}$$

Propositions of $\mathcal{P}$ can be combined through the $\wedge$, $\vee$ and $\neg$ connectives: $\forall p, q \in \mathcal{P}$,

$$v(p \wedge q) = v(p) \wedge v(q),$$

$$v(p \vee q) = v(p) \vee v(q),$$

$$v(\neg p) = \neg v(p).$$

Truth values are derived using the truth tables for $\wedge$, $\vee$ and $\neg$ below.

$\wedge$	T	M	F
T	T	M	F
M	M	M	F
F	F	F	F

$\vee$	T	M	F
T	T	T	T
M	T	M	M
F	T	M	F

$\neg$	
T	F
M	M
F	T

Table 5.1. Three Valued Logic Tables for $\wedge$, $\vee$ and $\neg$.

We extend the topological relations *in*, *overlap* and *disjoint* introduced in [CFvO93] to support three-valued topological relations. Our Three-valued Calculus Based Method (3-CBM) consists of the following topological relations:

Definition 5.2. The ***in*** 3-valued topological relation is defined as:

$$(X, in, Y) \iff (X \cap -Y = \emptyset) \wedge (X \cap Y \neq \emptyset)$$

Recall that $-Y$ represents the complement of Y, so that, $\lfloor -Y \rfloor = \mathcal{U} \setminus \lceil Y \rceil$ and $\lceil -Y \rceil = \mathcal{U} \setminus \lfloor Y \rfloor$.

Definition 5.3. The ***overlap*** 3-valued topological relation is defined as:

$$(X, overlap, Y) \iff X \cap -Y \neq \emptyset$$

Definition 5.4. The ***disjoint*** 3-valued topological relation is defined as:

$$(X, disjoint, Y) \iff X \cap Y = \emptyset$$

Thus, the topological relation (X, r, Y), where r is one of *in*, *overlap* and *disjoint* has a valuation $v((X, r, Y))$ which is either 'True', 'False' or 'Maybe'.

Figure 5.2 shows the topological relation decision tree of the 3-CBM, where $?r$ means $v((X, r, Y)) = M$ and $!r$ means $v((X, r, Y)) = T$. Furthermore, each branch of the tree is labeled 'T' (True), 'M' (Maybe) or 'F' (False) corresponding to the valuations of the proposition of the corresponding node.

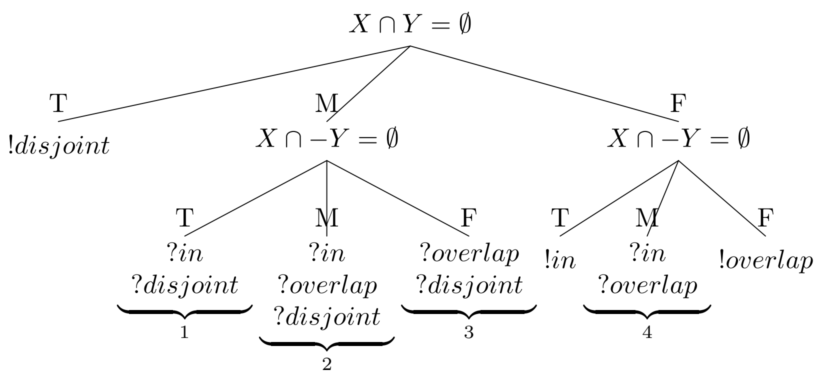


Figure 5.2. Topological Relations Decision Tree

Since the relations in the Three-valued Calculus Based Method are not crisp, more than one rough topological relation might evaluate to 'Maybe'. Thus, leaves of the decision tree labeled 1, 2, 3 and 4 correspond to more than one $?r$ rough topological relation, see Figure 5.2. Relations $!r$ evaluate to 'True' and exclude all other alternatives. The 7 leaves of the decision tree describe distinct topological relations, each of them can be described by a combination of the 3 topological relations *in*, *overlap* and *disjoint*.

Note that if the rough sets $(\lfloor X \rfloor, \lceil X \rceil)$ and $(\lfloor Y \rfloor, \lceil Y \rceil)$ are crisp, i. e. $\lfloor X \rfloor = \lceil X \rceil$ and $\lfloor Y \rfloor = \lceil Y \rceil$, topological relations (X, r, Y) will always evaluate to either 'True' or 'False'; in this case the our Three-valued Calculus Based Method reduces to the standard Calculus Based Method introduced in [CFvO93].

5.5 Conclusion

We have seen that a rough classification has an equivalent parameterized crisp classification. This gives a formal basis for derivation of a parameterized error matrix.

We have also shown how to use topological relations between rough classifications in order to derive qualitative information when assessing accuracies of rough classifications.

Chapter 6

Information System Design

This chapter shows how to implement an information system as the one presented in the examples of Chapter 4 using a relational database management system (RDBMS).

Section 6.1 presents the basic relations that are required to represent indiscernibility. Section 6.2 shows how to approximate a relation in the database. Section 6.3 shows how to use approximations in queries to the information system. Section 6.4 presents a way to represent classifications using the relational model. Section 6.5 discusses the truth-functional issue of approximation operators in our information system.

6.1 Information Systems in the Relational Model

We use the SQL relational language of the PostgreSQL database management system [Gro02] to implement our information system. See also [Mom01] for an introduction to the PostgreSQL RDBMS.

As seen in Example 4.2, the universe of our example information system is a set of features referred to by an identification number. Each feature is characterized by a set of colors that is represented in the table '`FeatureColors`' shown below. Note that each feature may be characterized by several colors, hence, the attribute '`colors`' below should be multi-valued. This is achieved by choosing the key of the table below to be a combination of both attributes '`id`' and '`colors`':

```
create table FeatureColors (
  -- feature id
  id integer,
  -- feature colors
  colors integer,
  primary key (id, colors)
);
```

The attribute ‘`colors`’ is of type integer as there is a code associated to each color, hence, we also define a table that describes the color codes as follows:

```
create table ColorCodes (
  colorId integer primary key,
  -- description
  color text
);
```

The feature table has been populated according to Example 4.2 and its content is listed below.

```
select id as feature, color as colors from FeatureColors, ColorCodes
  where colors = colorId
    order by id, colorId;

 feature |      colors
---------+------------------
       1 | slate gray
       1 | dark gray
       2 | lime green
       2 | yellow green
       2 | medium sea green
       3 | slate gray
       3 | dark gray
       4 | white
       5 | sky blue
       6 | white
       7 | light blue
       8 | white
       9 | lime green
       9 | yellow green
       9 | medium sea green
(15 rows)
```

For instance, feature of identity ‘1’ has attribute values ‘slate gray’ and ‘dark gray’ according to the table above.

If the information system had an additional attribute describing the features, an attribute on vegetation types for instance, then an additional table would be necessary as shown below.

```
create table FeatureVegetations (
  id integer,
  -- the attribute
  vegetations integer,
  primary key (id, vegetations)
);
```

This way, the attribute value of the feature of identity ‘`i`’ may be obtained through the following query:

```
select colors, vegetations
  from FeatureColors, FeatureVegetations
    where FeatureColors.id = i and
          FeatureColors.id = FeatureVagetations.id;
```

Note that the statement above returns a set of tuples instead of a tuple of sets, however, set results can be obtained by projecting on either attributes ‘`FeatureColors`’ or ‘`FeatureVegetations`’ individually.

6.2 Approximations

Here we show how to approximate subsets of the universe given an equivalence relation using the relational language SQL of the PostgreSQL RDBMS.

To model indiscernibility, we need an equivalence relation on the attribute values of ‘`colors`’. This equivalence relation can be defined as a Cartesian product stored in the following table:

```
create table EquivalentColors (
  colorsA integer,
  -- is equivalent to
  colorsB integer,
  primary key (colorsA, colorsB)
);
```

Here, the primary key declaration is not necessary, however, it avoids repetition of identical tuples. Elements that are equivalent to each other are stored as a tuple in ‘`EquivalentColors`’.

The equivalence relation of our running example may be displayed as shown below:

```
select ACol.color, BCol.color from
  EquivalentColors, ColorCodes as ACol, ColorCodes as BCol
    where colorA = ACol.colorId and colorB = BCol.colorId;
```

```
      color       |      color
------------------+------------------
 lime green       | medium sea green
 lime green       | lime green
 lime green       | yellow green
 yellow green     | medium sea green
 yellow green     | lime green
 yellow green     | yellow green
 medium sea green | medium sea green
 medium sea green | yellow green
 medium sea green | lime green
 sky blue         | sky blue
 sky blue         | light blue
 light blue       | sky blue
 light blue       | light blue
 slate gray       | dark gray
 slate gray       | slate gray
 dark gray        | slate gray
 dark gray        | dark gray
 white            | white
(18 rows)
```

Note that we have also performed a join with the color codes table in the query above to allow displaying color names instead of color codes.

Let us show how to compute approximations of attribute values.

We first define a function that returns the equivalence class of a given attribute value as follows:

```
create function equivColors(integer) returns setof integer
as '
  select colorB from EquivalentColors where colorA = $1;
' language 'sql';
```

Next we need a lower and an upper approximation operator on the attribute values of '`colors`'. The lower and upper approximation functions have an integer argument that determines the identity of the feature whose attribute value is to be approximated. The lower approximation function is defined as:

```
create function lowerColors(integer) returns setof integer
as '
  select color from ColorDomain
    where exists ((select equivColors(color))
      intersect (select colors from FeatureColors where id = $1))
        and
      not exists ((select equivColors(color))
        except (select colors from FeatureColors where id = $1))
' language 'sql';
```

The upper approximation function is defined as:

```
create function upperColors(integer) returns setof integer
as '
  select color from ColorDomain where
    exists ((select equivColors(color))
      intersect (select colors from FeatureColors where id = $1))
' language 'sql';
```

Above, ‘`ColorDomain`’ is a table that represents the domain of the attribute ‘`colors`. It is defined as:

```
create table ColorDomain (
  color integer primary key
);
```

Let us approximate the attribute value of the feature of identity ‘1’. The lower approximation is obtained as follows:

```
select lowerColor(1);

 lowercolors
-------------
           6
           7
```

The upper approximation is obtained as follows:

```
select upperColors(1);

 uppercolors
-------------
           6
           7
```

Since both lower and upper approximations are slate gray and dark gray, the feature of identity ‘1’ has an attribute value that is definable in our approximation space.

Let us print out the features and their approximate attribute values. The lower approximation of the attribute values are:

```
select A.id as feature, color as lower_approx from ColorCodes,
  (select distinct id, lowerColors(id) as i from FeatureColors) as A
     where colorId = A.i order by A.id;
```

```
 feature |   lower_approx
---------+------------------
       1 | dark gray
       1 | slate gray
       2 | lime green
       2 | medium sea green
       2 | yellow green
       3 | dark gray
       3 | slate gray
       4 | white
       6 | white
       8 | white
       9 | lime green
       9 | medium sea green
       9 | yellow green
(13 rows)
```

Note that the features of identity ‘5’ and ‘7’ are not listed in the resulting table above since the lower approximation of their attribute values are empty.
The upper approximation of the attribute values are:

```
select A.id as feature, color as upper_approx from ColorCodes,
  (select distinct id, upperColors(id) as i from FeatureColors) as A
     where colorId = A.i order by A.id;

 feature |   upper_approx
---------+------------------
       1 | dark gray
       1 | slate gray
       2 | lime green
       2 | yellow green
       2 | medium sea green
       3 | slate gray
       3 | dark gray
       4 | white
       5 | light blue
       5 | sky blue
       6 | white
       7 | sky blue
       7 | light blue
       8 | white
       9 | lime green
       9 | yellow green
       9 | medium sea green
(17 rows)
```

6.3 Queries

We show how to implement the query operation presented in Section 4.2.4. To do so we need to implement a lower and an upper approximation function that takes a set of color codes as argument and return the lower, respectively, the upper approximation of the set of colors. The lower approximation function is defined as:

```
create function lowerColorQuery(integer[]) returns setof integer
as '
  select color from ColorDomain where
    exists ((select equivColors(color))
      intersect (select * from toSet($1)))
      and
    not exists ((select equivColors(color))
      except (select * from toSet($1)))
' language 'sql';
```

The upper approximation function is defined as:

```
create function upperColorQuery(integer[]) returns setof integer
as '
  select color from ColorDomain where
    exists ((select equivColors(color))
      intersect (select * from toSet($1)))
' language 'sql';
```

The function '`toSet`' used above has as purpose to transform the array argument into a set. This is necessary as the current version of PostgreSQL (version 7.3) does not support set arguments in its procedural language (plpgsql).

We query features that have attribute values with the same upper and lower approximations as the set of colors given as argument. Hence the query function should be defined as follows:

```
create function colorQuery(integer[]) returns setof integer
as '
  select distinct id from FeatureColors where
    not exists ((select lowerColorQuery($1))
      except (select lowerColors(id)))
        and
    not exists ((select lowerColors(id))
      except (select lowerColorQuery($1)))
```

```
        and
    not exists ((select upperColorQuery($1))
      except (select upperColors(id)))
        and
    not exists ((select upperColors(id))
      except (select upperColorQuery($1)))
 '
language 'sql';
```

The function defined above queries elements of the universe by means of approximations of attribute values. We refer to this type of query as an *approximate query.*

As an example, we first query the features whose color attribute values have the same lower and upper approximation as the singleton value ‘4’, i. e. ‘sky blue’. The second query looks for features which are approximately ‘slate gray’ and ‘dark gray’, i. e. the set of color codes ‘6’ and ‘7’.

```
select * from colorQuery('{4}');

 colorquery
------------
         5
         7
(2 rows)

select * from colorQuery('{6, 7}');

 colorquery
------------
          1
          3
(2 rows)
```

Note that the examples above are based on a single attribute ‘`colors`’. In case we wish to represent queries on multiple attributes we need to implement similar relational tables and functions as for the attribute ‘`colors`’. Table 6.1 outlines the attribute specific implementations for ‘`colors`’.

In case of an additional attribute, for instance ‘`vegetations`’, all tables and functions specified in the Table 6.1 above should be specifically implemented for the ‘`vegetations`’ attribute. Queries would then be combinable using the intersection and the union operators. For instance, we could query the information system as follows:

```
(select colorQuery('{1, 2}'))
   intersect
(select vegetationQuery('{10}'));
```

Implementations for `colors`	Description
`FeatureColors`	stores feature's attribute values
`ColorDomain`	stores attribute domain
`ColorCodes`	stores descriptions of color codes
`EquivalentColors`	stores equivalence relation
`equivColors()`	in: color code, out: equivalence class
`lowerColors()`	in: feature id, out: lower colors value
`upperColors()`	in: feature id, out: upper colors value
`lowerColorQuery()`	in: array of color codes id, out: lower approx
`upperColorQuery()`	in: array color codes, out: upper approx
`colorQuery()`	in: array color codes, out: features

Table 6.1. Attribute Specific Implementations for Colors

or,

```
(select colorQuery('{4, 6}'))
   union
(select vegetationQuery('{9, 11}'))
```

6.3.1 Spatial Uncertainty

Let us note that the equivalence relation on the features of the information system can be obtained by the following query:

```
select distinct A.id as idA, B.id as idB into EquivalentFeatures
  from FeatureColors as A, FeatureColors as B where
    not exists ((select lowerColors(A.id))
      except (select lowerColors(B.id)))
      and
    not exists ((select lowerColors(B.id))
      except (select lowerColors(A.id)))
       and
    not exists ((select upperColors(A.id))
      except (select upperColors(B.id)))
      and
    not exists ((select upperColors(B.id))
      except (select upperColors(A.id)));
```

this leads to the creation of the '`EquivalentFeatures`' table show below.

```
select * from EquivalentFeatures;
 ida | idb
-----+-----
   1 |   1
   1 |   3
   2 |   2
   2 |   9
   3 |   1
   3 |   3
   4 |   4
   4 |   6
   4 |   8
   5 |   5
   5 |   7
   6 |   4
   6 |   6
   6 |   8
   7 |   5
   7 |   7
   8 |   4
   8 |   6
   8 |   8
   9 |   2
   9 |   9
(21 rows)
```

Now, we both have an equivalence relation on the attribute values and an induced equivalence relation on the features of the information system. Hence, we can also exhibit spatial uncertainty. For instance, the feature set '1', '5' and '7' is not definable in this approximation space (see Example 4.2). If we wish to query this feature set by means of colors descriptions we would need to compute a lower and an upper approximation as shown below.

```
-- lower approximation
select colorQuery('{4}');

 colorquery
------------
          5
          7
(2 rows)

-- upper approximation
select colorQuery('{4}') union select colorQuery('{6, 7}');
```

```
 colorquery
------------
          1
          3
          5
          7
(4 rows)
```

Thus, the feature set $X = \{‘1’, ‘5’, ‘7’\}$ is not definable and has to be described by its lower approximation: the set of features that are possibly ‘blue’ (color code ‘4’ for instance) and its upper approximation: the set of features that are possibly ‘blue’ *or* the set of features that are certainly ‘gray’:

$$\{‘5’, ‘7’\} \subseteq X \subseteq \{‘1’, ‘3’, ‘5’, ‘7’\}$$

6.4 Representing Classifications

We implement a table that will represent attribute classifications.

```
create table ColorClass (
  class integer primary key,
  colors integer[]
);
```

Each value of the attribute ‘`colors`’, as defined in Section 4.3, is an array that represents the attribute values that reclassify to a category determined by the value of the attribute ‘`class`’.

We continue the implementation of Example 4.4. The attribute classification table have been populated accordingly as follows:

```
select * from ColorClass

 class | colors
-------+-----------
     1 | {1, 2, 3, 4}
     2 | {5, 6, 7}
(2 rows)
```

According to the table, features that are characterized by a ‘green’ or a ‘sky blue’ color are reclassified to category ‘1’. Features that are characterized by a ‘light blue’ or a ‘gray’ color, on the other hand, are reclassified to category ‘2’. As in the example, categories ‘1’ and ‘2’ may represent, respectively, the presence and the absence of vegetations for instance.

Again, in our approximation space the colors ‘sky blue’ and ‘light blue’ are not discernible, hence, we require the use of approximate queries. The universal classification may be obtained by using lower and upper classification queries defined in Section 4.4. For this purpose, we implement the lower and the upper query operators ‘`lowerClassQuery(integer[])`’ and ‘`upperClassQuery(integer[])`’. These functions are implemented in PostgreSQL procedural language (plpgsql). The reason is that the argument of the functions has to be decomposed into a set of single-granular attribute values that form a set partition before the call to the query function ‘`colorQuery`’ can be made. This decomposition requires some side effect operations. The code of the functions has been omitted.

The universal classification is computed below. First, the lower approximation of the attribute classification is queried:

```
select class, lowerClassQuery(colors) as feature from ColorClass;

 class | feature
-------+---------
     1 |       2
     1 |       9
     2 |       1
     2 |       3
(4 rows)
```

As it appears above, feature ‘2’ and ‘9’ surely classify to category ‘1’ and features ‘1’ and ‘3’ surely classify to category ‘2’. To query features that possibly classify to a certain category we use the upper query as shown below:

```
select class, upperClassQuery(colors) as feature from ColorClass;

 class | feature
-------+---------
     1 |       2
     1 |       5
     1 |       7
     1 |       9
     2 |       1
     2 |       3
     2 |       5
     2 |       7
(8 rows)
```

Hence, features ‘2’, ‘5’, ‘7’ and ‘9’ possibly classify to category ‘1’. Features ‘1’, ‘3’, ‘5’ and ‘7’ possibly classify to category ‘2’. Also note that there is an uncertainty for the features ‘5’ and ‘7’ since they may classify to either category ’1’ or to category

'2'. Those features belong to the uncertain part of the universal classification, also called the *boundary* of the classification.

The query $\lfloor a = V_i \rfloor$ translates in our information system as:

```
select lowerClassQuery(colors)from ColorClass where class = 1;
```

and the query $\lceil a = V_i \rceil$ translates as:

```
select upperClassQuery(colors) from ColorClass where class = 1;
```

where a corresponds to the attribute '`colors`' and the set V_i to a set of color values such as $V_1 = \{\text{'1'}, \text{'2'}, \text{'3'}, \text{'4'}\}$ and $V_2 = \{\text{'5'}, \text{'6'}, \text{'7'}\}$.

As we noted in Section 4.4, for a given class i, the pair of queries $C_i = (\lfloor a = V_i \rfloor, \lceil a = V_i \rceil)$ is a rough set. Recall that the universal classification is the set of rough sets made up of the pairs of queries C_i for each category i. To query uncertain parts in the classification we need to perform intersections of rough sets. For instance, to query rough sets of the classification that possibly classify to both i and j, we need to perform the intersection $C_i \cap C_j$.

Below, we look at the truth-functionality issue of the intersection operator in our information system.

6.5 Information System and Truth Functionality

Truth functionality is an issue that we have discussed in Section 3.2. Let us illustrate this in our example information system. First, we consider approximation of sets of colors. The union of lower approximations:

```
select lowerColorQuery('{1,2,3,4}') union lowerColorQuery('{5,6,7}');

 lowercolorquery
-----------------
               1
               2
               3
               6
               7
(5 rows)
```

is not equal to the lower approximation of the union:

```
select lowerColorQuery('{1,2,3,4,5,6,7}');

 lowercolorquery
-----------------
               1
               2
               3
               4
               5
               6
               7
(7 rows)
```

Moreover, the intersection of upper the approximations:

```
select upperColorQuery('{1,2,3,4}') intersect lowerColorQuery('{5,6,7}');

 lowercolorquery
-----------------
               4
               5
(2 rows)
```

is not equal to the upper approximation of the intersection:

```
select upperColorQuery('{}');

 lowercolorquery
-----------------
(0 rows)
```

Now we look at universal classification.

If we consider each category apart in the classification of our running example, then, for category '1', the pair of queries:

```
select lowerClassQuery('{1,2,3,4}') as feature;

 feature
---------
      2
      9
(2 rows)
```

and

```
select upperClassQuery('{1,2,3,4}') as feature;

 feature
---------
      2
      9
      5
      7
(4 rows)
```

build up the rough set of features $C_1 = (\{2,9\}, \{2,9,5,7\})$ in the approximation space of equivalent features.

For category '2', the pair of queries:

```
select lowerClassQuery('{5,6,7}) as feature;

 feature
---------
      1
      3
(2 rows)
```

and

```
select upperClassQuery('{5,6,7}') as feature;

 feature
---------
      5
      7
      1
      3
(4 rows)
```

also build up a rough set of features $C_2 = (\{1,3\}, \{1,3,5,7\})$.

In order to compute features that possibly classify to category '1' or '2' we have to compute $C_1 \cap C_2$. This intersection does not lead to a rough set. This is the case since the attribute classification is defined using a set of class representatives that does not respect the truth-functionality rules of Section 3.2.3, i. e. the sets $V_1 = \{1,2,3,4\}$ and $V_2 = \{5,6,7\}$ are such that they both intersect the equivalence class $E = \{4,5\}$ but V_1 and V_2 do not intersect.

Let us look at the semantics of the attribute classification. The set V_1 indicates that a 'green' attribute value certainly belongs to category '1' and that a 'blue' attribute value possibly belongs to category '1':

$$\text{green} \xrightarrow{!} 1 \text{ and blue} \xrightarrow{?} 1$$

(see Section 4.3).

Similarly, V_2 indicates that a 'blue' attribute value possibly belongs to category '2' and that a 'gray' attribute value certainly belongs to category '2':

$$\text{blue} \xrightarrow{?} 2 \text{ and gray} \xrightarrow{!} 2$$

Let us modify the attribute classification as $V_1 = \{1,2,3,4\}$ and $V_2 = \{4,5,6\}$ for instance. This time we have that V_1 and V_2 intersect at E and they intersect each other at the equivalence class E, i. e. $V_1 \cap V_2 \cap E \neq \emptyset$. Hence, the truth functionality rules are respected for these two set representatives. Moreover, the semantics of the attribute classification is unchanged in the sense that the classification rules above still hold.

Let us look at intersection of classification queries. We query features that possibly classify to category ‘1’:

```
select upperClassQuery('{1,2,3,4}') as feature;

 feature
---------
       2
       9
       5
       7
(4 rows)
```

next, features that possibly classify to category ‘2’:

```
select upperClassQuery('{4,6,7}') as feature;

 feature
---------
       5
       7
       1
       3
(4 rows)
```

Hence, the intersection of the two queries above is the set features ‘5’ and ‘7’. Let us now query with $V_1 \cap V_2$, i. e. with the singleton color attribute value $\{4\}$:

```
select upperClassQuery('{4}') as feature;

 feature
---------
       5
       7
(2 rows)
```

Which is the same result as above. Moreover, it can easily be shown that the following equality holds:

$$(\lfloor a = V_1 \rfloor, \lceil a = V_1 \rceil) \cap (\lfloor a = V_2 \rfloor, \lceil a = V_2 \rceil) = (\lfloor a = V_1 \cap V_2 \rfloor, \lceil a = V_1 \cap V_2 \rceil)$$

which means that $C_1 \cap C_2$ is also a rough set.

Hence, if a given set of classification rules are based on an attribute classification whose elements respect truth-functionality rules then elements of the corresponding universal classification may be combined through the intersection and union operators.

6.6 Conclusion

We have shown how to represent approximation spaces in the relational model and we have introduced approximate queries. We have shown how approximate queries are used in universal classifications and we have discussed how truth-functionality affects the queries to the information system. The experiments performed on attribute classifications of our running example have confirmed that we can arrive at truth-functional operations provided that we put some restrictions on the elements of the classification.

It is relatively easy to add fuzzy capabilities to our information system. Each element of the information system may have a membership value to some fuzzy set X. This can be represented by a corresponding attribute in the relational table. The operators $\mu_{\underline{X}}$ and $\mu_{\overline{X}}$ are then implemented such that they return the minimum, respectively maximum, of the membership values of a given equivalence class.

Note that the implementation proposed here is entirely based on the standard relational algebra supported by most relational database systems. There may be another possible approach that is worthwhile investigating which is to implement the information system on top of a relational algebra that supports rough set technics. The idea of extending the relational algebra to support rough set technics has been presented by Beaubouef et al. in [BPB95].

The ability to store spatial data has not been implemented in the information system presented here. In Chapter 7 we present an earlier implementation that has been integrated into the AMOS II system [RJK00] to store spatial data. Porting the codes developed here to AMOS II is relatively easy when the necessary relational operators such as union and intersection are made available in the AMOS II system.

Chapter 7

Experiments

Michael F. Worboys states in [Wor94]: "it is the applications which are the life-blood of the field and give rationale to its more basic research."

We demonstrate the viability of the model developed in this thesis by presenting some real-life experiments.

Our implementation AMORose is presented in Section 7.1. In section 7.2 we present the rough classification analysis that was performed in our joint papers [AKO98, AKO00]. Section 7.3 presents the experiment on rough fuzzy classifications that has been presented in [AKO03].

7.1 Implementation Details

To implement our spatial information system we use the ROSE (Robust System Extension) library, a library of spatial operators developed by Güting et al. in [GDS95, GS93, GS95, Sch95]. This library has been fully integrated in an object-relational database system AMOS II (Active Mediator Object System). Detailed descriptions of the AMOS II system may be found in [FR98, RJK00] and [FJR$^+$99, Ris99]. This way, it is possible to manipulate geometric data through AMOSQL, the AMOS SQL-like relational query language.

We have integrated spatial uncertainty capabilities in our system AMORose, see [Ouk01] for a detailed presentation. Spatial uncertainty has been realized through our extension of the ROSE library called Rough ROSE [Ouk01].

7.2 An Example with Rough Classification

The goal of our experiment was to use the idea of rough classification to compare two vegetation maps, covering the same area of 1.8 by 2.3 square kilometers in Stockholm County, just north of Stockholm. The two maps had been produced for nature preservation tasks and different classification schemes were used to delineate vegetation categories on a categorical map sheet in 1:15000 scale.

In the experiment we considered the two layers as two different representations of the same area. We started out with two vegetation maps which provided two crisp vegetation classification layers called 'veg9' and 'veg35'. Building on vegetation concepts that were introduced by Påhlsson in [Påh72], 'veg9' was classified using moisture and nutrient status as classification basis, giving nine different vegetation classes for the experiment area. The vegetation map 'veg35' used a Nordic classification system described by Påhlsson in [Påh95], giving 35 different vegetation classes for the experiment area. The class descriptions are given in [AKO00]. To compare the two, we needed to reclassify one of them; for obvious reasons, we have chosen 'veg35'.

Reclassification of 'veg35' into 'veg9' would be a straightforward task if a one-to-one or many-to-one correspondence between the two classification systems existed. As this is not the case, we used the idea of rough classification to represent the uncertainty in the reclassification operation. Rules to reclassify from the classification system given by Påhlsson [Påh95] to the one given by Påhlsson [Påh72] were constructed using both the guidelines given in [Påh95] and our own knowledge about the association between the different vegetation classes. These classification rules are given in [AKO00]. Note that here we use spatial uncertainty to represent the reclassification of 'veg35' into the nine classes of 'veg9'.

Figure 7.2 outlines the analysis described above. In the figure, single boxes represent crisp classifications and double boxes represent rough classifications.

The purpose of the analysis below is to compute the overall accuracy of 'veg9' and 'veg35' which has been derived in [AKO98, AKO03].

The first step in the analysis is to populate the spatial database. We omit the code for populating the database. Each layer type 'veg35' and 'veg9' corresponds to a relation in our database. Each of them consist of a set of non-overlapping polygons, so each can be viewed as a tessellation. The Layer 'veg35', denoted `veg35` in the AMOSQL code below, is a subtype of `layer`,

```
create type veg35 subtype of layer;
```

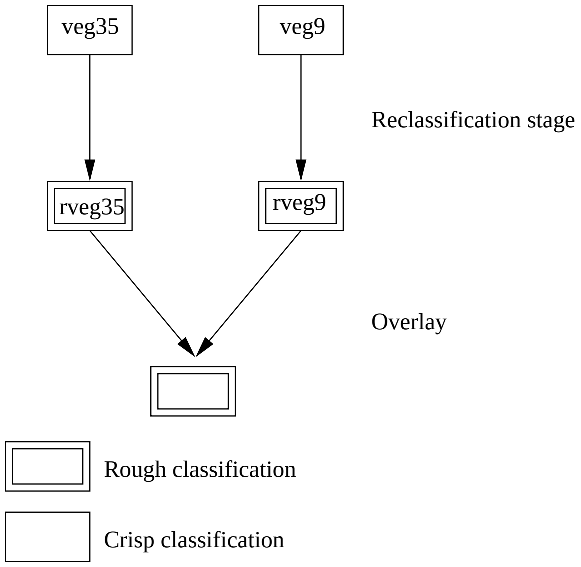
The layers `veg35` and `veg9` cover exactly the same area of size `total_area` which is computed as follows:

```
set :total_area =
          select sumagg(select area(reg(v35)) from veg35 v35);
```

The `sumagg` AMOSQL operator above is the sum aggregation operator on a bag of numbers.

The second step of the analysis is to create a relation that will represent rough classifications:

```
create type classification properties (class integer key,
                                       rreg rregions)
```



In a rough classification the key attribute `class` uniquely identifies a given class. In AMORose, spatial uncertainty is represented through objects of type rough regions `rregions`. Because of spatial uncertainty, rough regions may have parts that belong to more than one class.

There are two rough classifications:

```
create type rveg35 subtype of classification;
create type rveg9 subtype of classification;
```

Note that 'rveg9' is the reference crisp classification and is a particular case of rough classification.

The reclassification rules from the classification scheme of 'veg35' to the classification scheme of 'veg9' are of the form $a \longrightarrow i \vee j$. This rule means that a class a in the classification scheme of 'veg35' is either of attribute i or attribute j in the classification scheme of 'veg9'.

The classification 'rveg35' represents the reclassification of 'veg35' into the classification scheme of 'veg9'.

Once ‘rveg35’ and ‘rveg9’ have been populated through reclassification, we compute the overall accuracy of ‘rveg35’ compared to ‘rveg9’:

```
set :max_area = select sumagg(
select Maxarea(Intersection(reg(v35), reg(v9)))
       from rveg35 v35, rveg9 v9
                   where class(v35) = class(v9));
```

The code above computes the area of the overlay of matching upper approximations of ‘rveg35’ and ‘rveg9’,

```
set :min_area = select sumagg(
select Minarea(Intersection(reg(v35), reg(v9)))
       from rveg35 v35, rveg9 v9
                   where class(v35) = class(v9));
```

computes the area of the overlay of matching lower approximations of ‘rveg35’ and ‘rveg9’. The maximum overall accuracy is given by:

```
set :max_accuracy = :max_area/:total_area;
```

The minimum overall accuracy is given by:

```
set :min_accuracy = :min_area/:total_area;
```

Finally, the overall accuracy of ‘rveg35’ compared to the reference classification ‘rveg9’ is the interval:

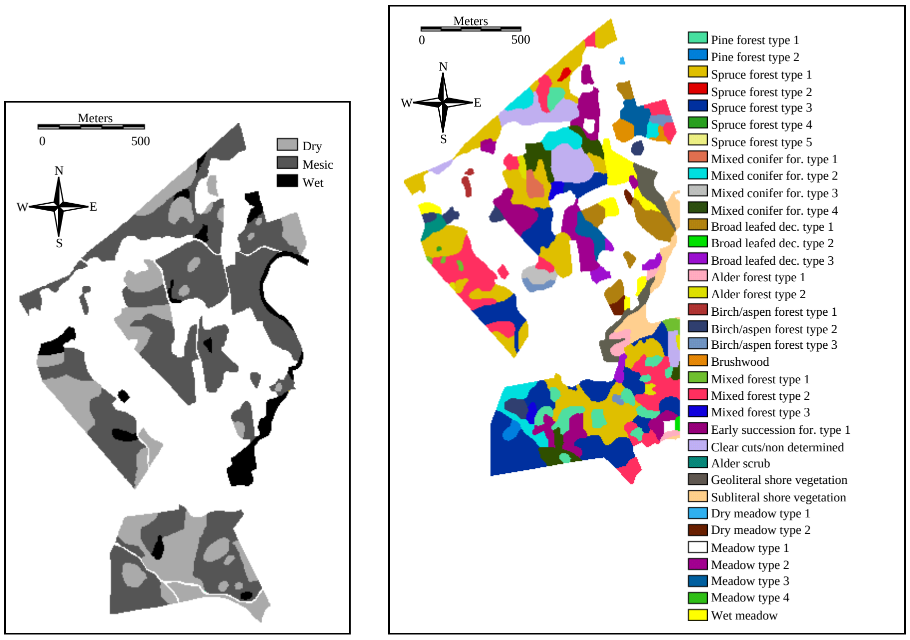
```
[min_accuracy, max_accuracy] = [0.039, 0.461]
```

This interval means that the overall accuracy of ‘rveg35’ is at least ‘`min_accuracy`’ and at most ‘`max_accuracy`’.

7.3 Rough Fuzzy Experiment

The experiment presented here has been described in our joint paper [AKO03] in which we demonstrate the viability of rough fuzzy classification. The raw data used here is the same as in Section 7.2 above, however, some other classification schemes are considered. Moreover, this experiment was carried out using different software.

The purpose of the experiment is to compare two vegetation maps ‘veg3’ and ‘veg35’, shown in figure 7.1. These maps were taken from two separate investigations covering the same area [ELP71, AW90] and have been produced for nature preservation tasks. However, ‘veg3’ and ‘veg35’ use different vegetation classification schemes. The map ‘veg3’, see Figure 7.1(a), uses 3 classes: wet, mesic and dry vegetation. ‘veg35’, on the other hand, uses 35 classes, see Figure 7.1(b). Both ‘veg3’ and ‘veg35’ were digitized by scanning and segmentation into two GIS raster images with pixel values representing vegetation class labels. Data entry and all



(a) The veg3 map.

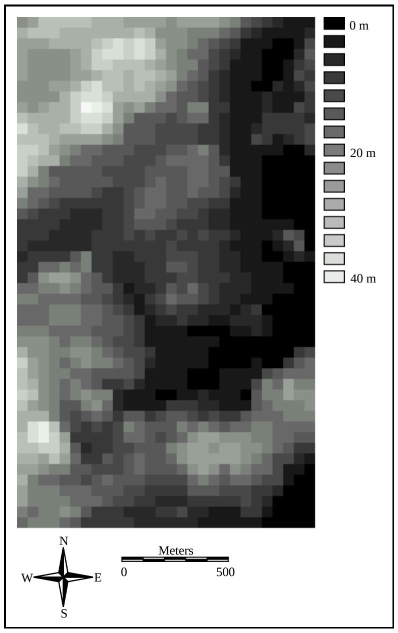
(b) The veg35 map.

Figure 7.1. Original maps used in the rough-fuzzy experiment.

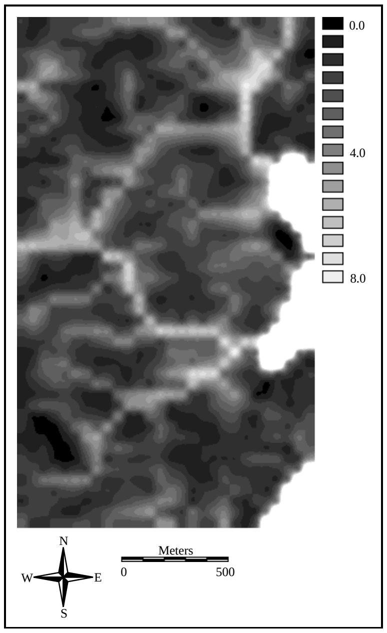
subsequent GIS operations were performed using the IDRISI for Windows v.2.0 software.

In order to compare ‘veg3’ and ‘veg35’, as in [AKO00], we proceed by reclassifying ‘veg35’ into a rough classification, called ‘Rveg35’, using the classification scheme of ‘veg3’: ‘wet’, ‘mesic’ and ‘dry’. The rules for reclassification were established using domain knowledge. Each original ‘veg35’ class was evaluated against the target classification system of ‘veg3’ and translated into rough classification rules, see Table 7.1. The resulting rough classification ‘Rveg35’ is shown in Figure 7.4. The overlap and the crispness measures for ‘Rveg35’ are $\mathcal{O} = 0.64$ and $\mathcal{C} = 0.41$, respectively.

The ‘veg3’ classification system is based upon vegetation moisture thus additional information about soil water content may resolve some of the indiscernibility that was a result of the rough reclassification of ‘veg35’. Such information might be provided by a digital elevation model ‘dem’ over the area covered by ‘veg3’. One commonly used source of information on soil water content is the topographically based wetness index, w [MGL91]. This index is calculated for each cell in an elevation image using the following formula: $w = \ln(A_s / \tan\beta)$ where A_s is specific catchment area defined as the upslope area draining across a unit width of contour, and β is the slope of the cell. Thus, a wetness index image was calculated from an original digital elevation model from Swedish Geographic Data (GSD) provided by the Swedish National Land Survey, using the TAPES-G software [GW96]. The used digital elevation model had a spatial resolution of $50 \times 50\text{m}^2$. The calculated wetness index values were interpolated over a $5 \times 5\text{m}^2$ resolution image. The purpose for this was to transform the original $50 \times 50\text{m}^2$ resolution into the map data at $5 \times 5\text{m}^2$ resolution. The accuracy of the output images is of course questionable, but the objective of this experiment is only to perform a feasibility test of rough-fuzzy geographic data integration, not to produce a fully accurate representation. The DEM together with the derived wetness image are displayed in Figure 7.2.



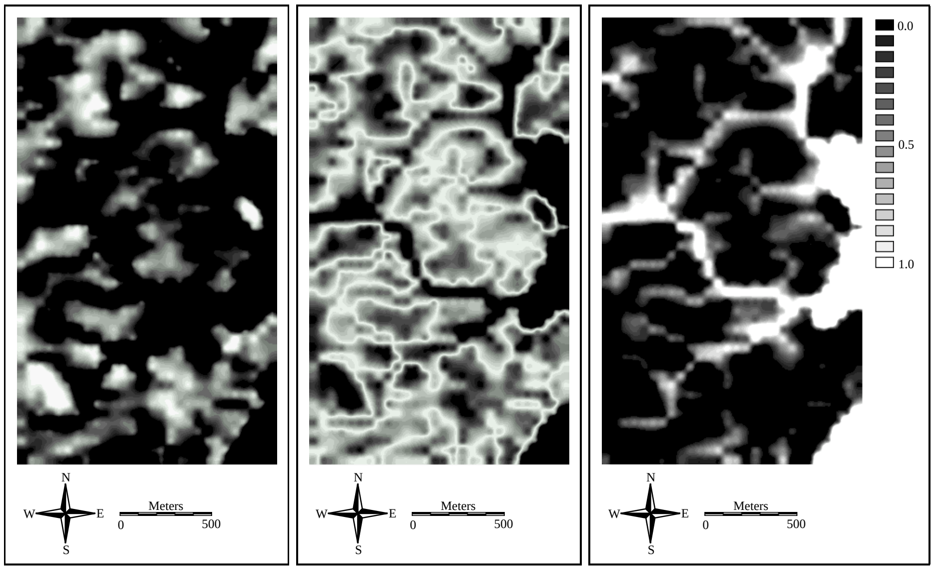
(a) Original digital elevation model.



(b) Wetness index image.

Figure 7.2. Calculating the wetness index.

The fuzzy membership functions used to transform the wetness image into fuzzy information were established using the semantic import approach (cf. [BM98]). Note that the fuzzy concept used here is *wetness*. This method is suitable as we have a fairly good qualitative idea of how to group data. The two major issues in the semantic import approach are the choice between linear, sinusoidal or some other function defining the class membership and the definition of the transition zone limits and widths. To make things simple we chose linear transition functions and to support the establishment of transition zone limits and widths, information from the older vegetation map was used as a guiding reference. The established membership functions displayed in Figure 7.3 were used to produce the fuzzy classification ‘Fdem’ from the digital elevation model ‘dem’. ‘Fdem’ consists of three fuzzy classes μ_{X_i}, $i \in \{\text{dry}, \text{mesic}, \text{wet}\}$. The three pictures of Figure 7.3 show the membership values at every pixel location in each of the three fuzzy classes, ‘dry’, ‘mesic’ and ‘wet’.

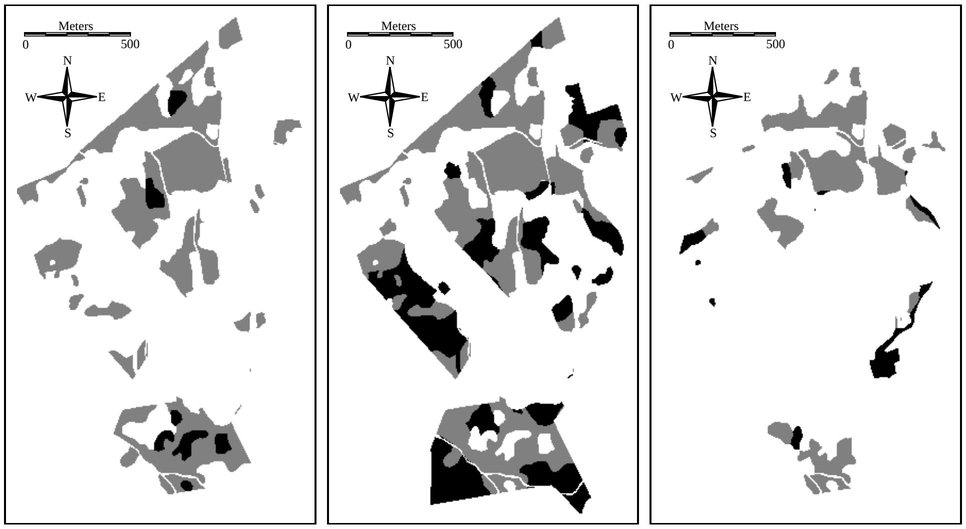


(a) dry (b) mesic (c) wet

Figure 7.3. The fuzzy classification ‘Fdem’. Membership values range between full membership in white areas and no membership in black areas.

We converted the crisp classification ‘veg3’, the rough classification ‘Rveg35’ and the fuzzy classification ‘Fdem’ into rough fuzzy classifications ‘RFveg3’, ‘RFveg35’ and ‘RFdem’ respectively. The resulting rough fuzzy classifications hold the same information as the original crisp, rough and fuzzy classifications, only the representation has been changed. For instance, the rough fuzzy classification ‘RFdem’ is now represented by a set of six images in which only three differ from each other, i. e. ‘RFdem’ still looks exactly as ‘Fdem’ in Figure 7.3.

In order to resolve some of the uncertainty present in the data, ‘RFveg35’ and ‘RFdem’ have been combined through the intersection operation. The resulting RF-classification is called ‘RFresult’, i.e. RFresult = ‘RFveg35’ $\cap$ ‘RFdem’. ‘RF-



(a) Dry vegetation. (b) Mesic vegetation. (c) Wet vegetation.

Figure 7.4. The veg35 layer after reclassification. Lower approximations in black; areas of uncertainty in grey.

result' is shown in Figure 7.5 as an image for each of $\mu_{\utilde{X_{\text{dry}}}}$, $\mu_{\utilde{X_{\text{mesic}}}}$, $\mu_{\utilde{X_{\text{wet}}}}$, $\mu_{\widetilde{X_{\text{dry}}}}$, $\mu_{\widetilde{X_{\text{mesic}}}}$ and $\mu_{\widetilde{X_{\text{wet}}}}$. 'RFresult' have a roughness measure of $r = 0.67$ and an overlap measure of $\mathcal{O} = 0.18$. Note that the overlap measure has decreased considerably.

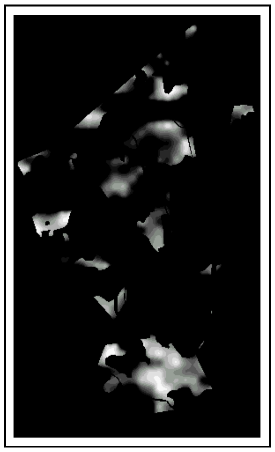
7.4 Conclusion

We have show that it is possible to use rough and rough fuzzy classification in real-life experiments.

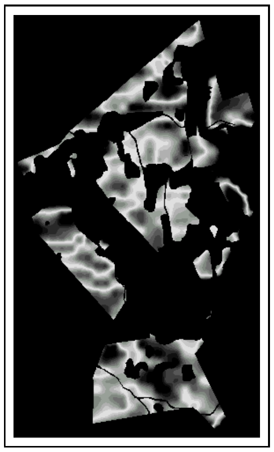
These experiments have been performed in an early stage of the research and there has not been any attempt to model the classifications presented here using the query model presented in Chapter 4 and the implementation of Chapter 6, however, it would be interesting to do so. In order to perform such experiment some further investigation is needed on how to represent a knowledge about some reclassification rules by using attribute classifications.

Original classification I	Rough classification R
1 : Pine forest type 1	{Dry}
2 : Pine forest type 2	{Dry, mesic}
3 : Spruce forest type1	{Dry, mesic}
4 : Spruce forest type2	{Mesic, wet}
5 : Spruce forest type3	{Mesic}
6 : Spruce forest type4	{Wet}
7 : Spruce forest type5	N/A
8 : Mixed conifer forest type 1	{Dry}
9 : Mixed conifer forest type 2	{Dry, mesic}
10 : Mixed conifer forest type 3	{Mesic}
11 : Mixed conifer forest type 4	{Dry, mesic, wet}
12 : Broad leafed deciduous forest type 1	{Mesic}
13 : Broad leafed deciduous forest type 2	N/A
14 : Broad leafed deciduous forest type 3	{Mesic}
15 : Alder forest type 1	{Wet}
16 : Alder forest type 2	{Wet}
17 : Birch/aspen forest type 1	{Dry, mesic}
18 : Birch/aspen forest type 2	{Mesic, wet}
19 : Birch/aspen forest type 3	{Dry, mesic}
20 : Brushwood	{Mesic, wet}
21 : Mixed forest type 1	{Dry, mesic}
22 : Mixed forest type 2	{Mesic}
23 : Mixed forest type 3	{Wet}
24 : Early succession forest type 1	{Dry, mesic, wet}
25 : Clear cuts/non determined	{Dry, mesic, wet}
26 : Alder scrub	{Wet}
27 : Geoliteral shore vegetation	{Wet}
28 : Subliteral shore vegetation	N/A
29 : Dry meadow type 1	N/A
30 : Dry meadow type 2	{Dry, mesic}
31 : Meadow type 1	{Dry, mesic}
32 : Meadow type 2	{Dry, mesic}
33 : Meadow type 3	{Mesic}
34 : Meadow type 4	N/A
35 : Wet meadow	{Mesic, wet}

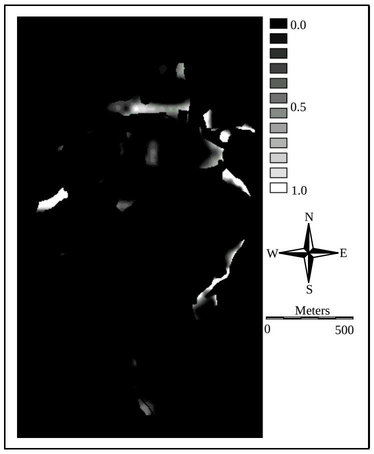
Table 7.1. Reclassification rules.



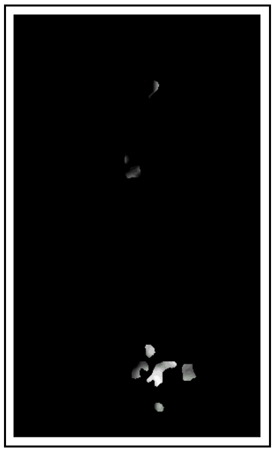
(a) Dry vegetation; upper approximation.



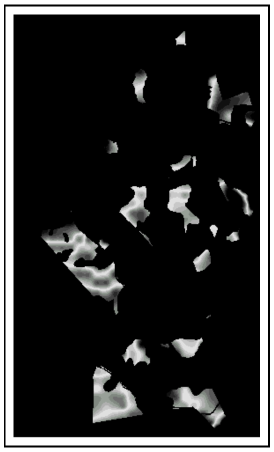
(b) Mesic vegetation; upper approximation.



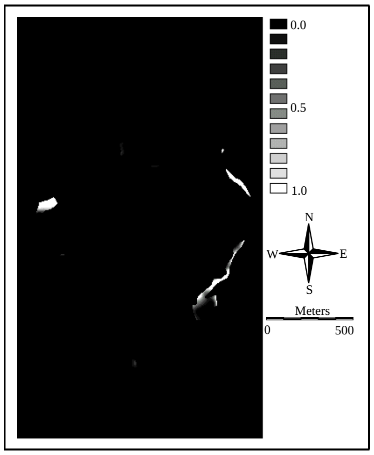
(c) Wet vegetation; upper approximation.



(d) Dry vegetation; lower approximation.



(e) Mesic vegetation; lower approximation.



(f) Wet vegetation; lower approximation.

Figure 7.5. The resulting rough-fuzzy classification.

Chapter 8

Conclusion

We have presented a theoretical model for a language that supports queries by means of uncertain descriptions. This model has also been implemented in an SQL-based relational query language to demonstrate the feasibility of the idea. We have shown how to define uncertain classifications using our uncertain-query language and we have developed some measures to assess accuracies of classifications. Finally, we have shown the viability of uncertain classifications by performing some experiments.

One of the main contributions of this thesis is in the area of classifications. Based on rough sets and fuzzy sets, we have given an underlying theory to explain how classifications may be performed.

Assume for instance that you are interested in classifying a certain area into 'forest' and 'non-forest' classes. This classification is often uncertain as it may not be easy to draw a boundary that sharply divides forest and non-forest between two distinct areas. Instead, one may be able to draw a sharp boundary somewhere between the perimeter of an area that is certainly forest and that of an area which is certainly not forest. Hence, we arrive at a classification with two boundaries which reminds us of a rough set. However, the equivalence relation is not known in this case. This thesis proposes a solution to this issue by using approximate queries. In our model, one may exhibit the equivalence relation by querying an information system by means of uncertain descriptions.

The classifications that are presented in this thesis are based on information systems whose elements have some particular attribute values. The particularity of the attribute values in question is that they are approximated by at most one equivalence class in the corresponding approximation space. Note that in this thesis we have introduced the concept of single-granular information systems to refer to this type of information system. This has been done to simplify the discourse. Nevertheless, further research is needed to find other types of classifications that are based on more general information systems. This may require taking into consideration the different interpretations of multi-valued attributes.

We conjecture that there are new opportunities for query optimization in database systems that implement uncertain queries. Query optimization is known to be an NP-hard problem [IK84]. The same complexity holds for distributed query optimization [ÖV99], i. e. for optimizing queries that are executed on different sites. This latter case of query optimization is especially interesting to study in conjunction with approximate queries. The reason is that the heuristics used in producing a near optimal query plan often takes total communication cost into account. One of the particularities of rough sets is that they can be used to *simplify* data by describing them with an upper and a lower approximation. By transmitting only lower and upper approximations of queries, one may minimize transmission costs. This may be achieved by transmitting only names of equivalence classes.

References

[AKO98] O. Ahlqvist, J. Keukelaar, and K. Oukbir. Using rough classification to represent uncertainty in spatial data. In P. Firns, editor, *Proceedings of the SIRC Colloquium*, pages 1–9, 1998.

[AKO00] O. Ahlqvist, J. Keukelaar, and K. Oukbir. Rough classification and accuracy assessment. *International Journal of Geographical Information Science*, 14(5):475–496, 2000.

[AKO03] O. Ahlqvist, J. Keukelaar, and K. Oukbir. Rough and fuzzy geographical data integration. *International Journal of Geographical Information Science*, 17(3):223–234, 2003.

[AW90] O. Ahlqvist and P. Wiborn. *Tranviks Naturreservat – Naturinventering med förslag till skötselplan.* Norrtälje Kommun, 1990. In Swedish.

[BM98] P. A. Burrough and R. A. McDonnell. *Principles of Geographical Information Systems.* Oxford University Press, 1998.

[BP01] T. Beaubouef and F. Petry. Vagueness in spatial data: Rough set and egg-yolk approaches. In L. Monostori, J. Váncza, and M. Ali, editors, *Engineering of Intelligent Systems, Proceedings of IEA/AIE 2001*, number 2070 in Lecture Notes in Artificial Intelligence, pages 367–373, Berlin, 2001. Springer-Verlag.

[BPB95] T. Beaubouef, F. Petry, and B. Buckles. Extension of the relational database and its algebra with rough set techniques. *Computational Intelligence*, 11(2), 1995.

[CF96] E. Clementini and P. Di Felice. An algebraic model for spatial objects with undeterminate boundaries. In P. A. Burrough and A. U. Frank, editors, *Geographic Objects with Indeterminate Boundaries*, volume 2 of *GISDATA*, pages 155–169, London, 1996. Taylor & Francis.

[CF97] E. Clementini and P. Di Felice. Approximate topological relations. *International Journal of Approximate Reasoning*, 16(2):173–204, 1997.

[CFvO93] E. Clementini, P. Di Felice, and P. van Oosterom. A small set of formal topological relationships suitable for end-user interaction. In D. Abel and B. C. Ooi, editors, *Third International Symposium on Large Spatial Databases, SSD '93*, volume 692 of *Lecture Notes in Computer Science*, pages 277–295, Berlin, June 1993. Springer-Verlag.

[CG96] A. G. Cohn and N. M. Gotts. The 'egg-yolk' representation of regions with indeterminate boundaries. In P. A. Burrough and A. U. Frank, editors, *Geographic Objects with Indeterminate Boundaries*, volume 2 of *GISDATA*, pages 171–187, London, 1996. Taylor & Francis.

[Cod70] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[Con91] R. G. Congalton. A review of assessing the accuracy of classifications of remotely sensed data. *Remote Sensing Environment*, 47:35–46, 1991.

[DG00] I. Düntsch and G. Gediga. Logical and algebraic techniques for rough set data analysis. In Lech Polkowski, Shusaku Tsumoto, and Tsau Young Lin, editors, *Rough Set Methods and Applications*, pages 521–544. Physica Verlag, 2000.

[DGO01] I. Düntsch, G. Gediga, and E. Orlowska. Relational attribute systems. *International Journal of Human-Computer Studies*, 55(3):293–309, 2001.

[DP90] D. Dubois and H. Prade. Rough fuzzy sets and fuzzy rough sets. *International Journal of General Systems*, 17:191–209, 1990.

[DP92] D. Dubois and H. Prade. Putting rough sets and fuzzy sets together. In R. Slowinski, editor, *Intelligent Decision Support – Handbook of Advances and Applications of the Rough Set Theory*, pages 203–232. Kluwer Academic Publishers, Dordrecht, Boston, London, 1992.

[Ege94] M. J. Egenhofer. Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):86–95, 1994.

[ELP71] E. Edberg, T. Lindberg, and M. Petterson. *Ängsö Nationalpark och Tranvik – vegetation och fågelfauna*. Rotobeckman, Stockholm, Sweden, 1971. In Swedish.

[FJR$^+$99] S. Flodin, V. Josifovski, T. Risch, M. Sköld, and M. Werner. *AMOS II User's Guide*. EDSLAB, Department of Computer Science Linköping University, August 1999.

[FR98] G. Fahl and T. Risch. *AMOS2 Introduction*. EDSLAB, Department of Computer Science, Linköping University, September 1998.

[GDS95] R. H. Güting, T. De Ridder, and M. Schneider. Implementation of the ROSE algebra: Efficient algorithms for realm-based spatial data types. In *Lecture Notes in Computer Science*, volume 951 of *4th Int. Symp. on Advances in Spatial Databases (SSD)*, pages 216–239, 1995.

[Goo93] M. F. Goodchild. Data models and data quality: Problems and prospects. In M. F. Goodchild, B. O. Banks, and L. T. Steynaert, editors, *Visualization in Geographical Information Systems*, pages 141–149. John Wiley, New York, 1993.

[Gro02] The PostgreSQL Global Development Group. *PostgreSQL 7.3 Documentation*, 2002.

[GS93] R. H. Güting and M. Schneider. Realms: A foundation for spatial data types in database systems. In D. Abel and B. C. Ooi, editors, *Third International Symposium on Large Spatial Databases, SSD '93*, volume 692 of *Lecture Notes in Computer Science*, pages 14–35, Berlin, June 1993. Springer-Verlag.

[GS95] R. H. Güting and M. Schneider. Realm-based spatial data types: The ROSE algebra. *The VLDB Journal*, 4(2):243–286, April 1995.

[GW92] M. Gehrke and E. Walker. On the structure of rough sets. *Bull. Polish Acad. Sci. Math.*, 40:235–245, 1992.

[GW96] J. C. Gallant and J. P. Wilson. TAPES-G: A grid-based terrain analysis program for the environmental sciences. *Computers and Geosciences*, 22(7):713–722, 1996.

[HS93] Z. Huang and P. Svensson. Neighborhood query and analysis with GeoSAL, a spatial database language. In *LNCS 692*. Springer-Verlag, 1993.

[HSH92] Z. Huang, P. Svensson, and H. Hauska. Solving spatial analysis problems with GeoSAL, a spatial query language. In *Proceedings of the 6th Int. Working Conf. on Scientific and Statistical Database Management*, 1992.

[IK84] T. Ibaraki and T. Kameda. On the optimal nesting order for computing n-relation joins. *ACM Transactions on Database Systems*, 9(3):482–502, 1984.

[Iwi87] T. B. Iwinski. Algebraic approach to rough sets. In *Bulletin of the Polish Academy of Sciences, Mathematics*, volume 35, pages 673–683. A. Lasota, 1987.

[Keu02] J. Keukelaar. *Topics in soft computing.* PhD thesis, Royal Institute of Technology, Stockholm, 2002.

[Kle57] S.C. Kleene. *Introduction to Metamathematics.* Van Nostrand, 1957.

[KR96] M. Kryszkiewicz and H. Rybinski. Reducing information systems with uncertain attributes. In Z. W. Rás and M. Michalewicz, editors, *Proceedings of the Ninth International Symposium on Foundations of Intelligent Systems*, volume 1079 of *LNAI*, pages 285–294. Springer, 1996.

[LC95] F. Lehmann and A. G. Cohn. The eggyolk reliability hierarchy: Semantic data integration using sorts with prototypes. In *Proceedings of the third international conference on Information and knowledge management*, pages 272–279. ACM Press, 1995.

[Lin01] F. Lindström. *Världens dåligaste språk: tankar om språket och människan.* Månpocket, 2001. In Swedish.

[MGL91] I. D. Moore, R. B. Grayson, and A. R. Ladson. Digital terrain modelling: A review of hydrological, geomorphological and biological applications. *Hydrologic Processes*, 5(1):3–30, 1991.

[Mom01] B. Momjian. *PostgreSQL: introduction and concepts.* Addison-Wesley, Reading, MA, USA, 2001.

[MT99] V. W. Marek and M. Truszczyński. Contributions to the theory of rough sets. *FUNDINF: Fundamenta Informaticae*, 39(4):389–409, 1999.

[OKA01] K. Oukbir, J. Keukelaar, and O. Ahlqvist. Rough point set topological relations. Nada Report, ref. number TRITA-NA-0111, 2001.

[Ouk97] K. Oukbir. Amorose, a realm based spatial database system. In *Proceedings of ScanGIS*, 1997.

[Ouk01] K. Oukbir. A database query language for uncertain spatial data. Department of Numerical Analysis and Computer Science (Nada), KTH. Licentiate Thesis, ISBN 91-7283-134-0, 2001.

[ÖV99] M. Tamer Özsu and P. Valduriez. *Principles if Distributed Database Systems.* Prentice Hall, second edition, 1999.

[Påh72] L. Påhlsson. *Översiktlig Vegetationsinventering.* Statens Naturvårdsverk, Stockholm, 1972. In Swedish.

[Påh95] L. Påhlsson. *Vegetationstyper i Norden.* Number 1994:665 in Tema Nord. Nordic Council of Ministers, Copenhagen, 1995. in Swedish.

[Paw82] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(5):341–356, 1982.

[Paw85] Z. Pawlak. Rough sets and fuzzy sets. *Fuzzy Sets and Systems*, 17(1):99–102, 1985.

[Paw91] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht, 1991.

[Paw96] Z. Pawlak. Rough sets, rough relations and rough functions. *FUNDINF: Fundamenta Informaticae*, 27, 1996.

[Pol02] L. Polkowski. Rough sets, mathematical foundations. In J. Kacprzyk, editor, *Advances in Soft Computing*, chapter 14. Physica-Verlag, 2002.

[Ris99] T. Risch. *AMOS II External Interfaces*. EDSLAB, Department of Computer Science Linköping University, April 1999.

[RJK00] T. Risch, V. Josifovski, and T. Katchaounov. AMOS II concepts. Technical report, Uppsala Database Laboratory (UDBL), June 2000.

[RS01] A. J. Roy and J. G. Stell. Spatial relations between indeterminate regions. *International Journal of Approximate Reasoning*, 27(3):205–234, 2001.

[SC03] S. Shekhar and S. Chawla. *Spatial Databases: A Tour*. Prentice Hall, 2003.

[Sch95] M. Schneider. *Spatial Data Types for Database Systems*. PhD thesis, Fachbereich Informatik, FernUniversität Hagen, 1995.

[SW98] J. G. Stell and M. F. Worboys. Stratified map spaces. In T. Poinker and N. Chrisman, editors, *Proceedings of the 8th International Symposium on Spatial Data Handling*, pages 180–189. Taylor and Francis, 1998.

[Thi98] H. Thiele. Fuzzy rough sets versus rough fuzzy sets– an interpretation and a comparative study using concepts of modal logics. Technical Report CI-30/98, University of Dortmund Dept. of Computer Science, 1998.

[WC01] M. F. Worboys and E. Clementini. Integration of imperfect spatial information. *Journal of Visual Languages and Computing*, 12:61–80, 2001.

[Wor94] M. Worboys, editor. *Innovations in GIS 1*, chapter 3. Taylor & Francis, London, UK., 1994.

[Yao93] Y. Y. Yao. Interval-set algebra for qualitative knowledge representation. In *Proc. 5th international Conference on Computing and Information*, pages 370–374, 1993.

[Yao96] Y. Y. Yao. Two views of the theory of rough sets in finite universes. *International Journal of Approximate Reasoning*, 15:291–317, 1996.

[Yao97] Y. Y. Yao. Combination of rough and fuzzy sets based on α-level sets. In T. Y. Lin and N. Cercone, editors, *Rough Sets and Data Mining: Analysis for Imprecise Data*, volume 104, pages 301–321. Kluwer Academic Publishers, 1997.

[YL96] Y. Y. Yao and X. Li. Comparison of rough-set and interval-set models for uncertain reasoning. *FUNDINF: Fundamenta Informatica*, 27, 1996.

[YWL97] Y. Y. Yao, S. K. M. Wong, and T. Y. Lin. A review of rough set models. In T. Y. Lin and N. Cercone, editors, *Rough Sets and Data Mining: Analysis for Imprecise Data*, pages 47–75. Kluwer Academic Publishers, Boston, 1997.

[Zad65] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.