Reasoning with Rough Sets and
Paraconsistent Rough Sets

Copyright © 2010 Aída Vitória unless otherwise noted
aida.vitoria@itn.liu.se

Department of Science and Technology, Linköping University
SE-601 74 Norrköping, Sweden

ISBN 978-91-7393-411-4
ISSN 0345-7524

This thesis is available online through Linköping University Electronic Press:
www.ep.liu.se

Printed by LiU-Tryck, Linköping, Sweden
Abstract

This thesis presents an approach to knowledge representation combining rough sets and paraconsistent logic programming.

The rough sets framework proposes a method to handle a specific type of uncertainty originating from the fact that an agent may perceive different objects of the universe as being similar, although they may have different properties. A rough set is then defined by approximations taking into account the similarity between objects. The number of applications and the clear mathematical foundation of rough sets techniques demonstrate their importance. Most of the research in the rough sets field overlooks three important aspects. Firstly, there are no established techniques for defining rough concepts (sets) in terms of other rough concepts and for reasoning about them. Secondly, there are no systematic methods for integration of domain and expert knowledge into the definition of rough concepts. Thirdly, some additional forms of uncertainty are not considered: it is assumed that knowledge about similarities between objects is precise, while in reality it may be incomplete and contradictory; and, for some objects there may be no evidence about whether they belong to a certain concept.

The thesis addresses these problems using the ideas of paraconsistent logic programming, a recognized technique which makes it possible to represent inconsistent knowledge and to reason about it. This work consists of two parts, each of which proposes a different language. Both languages cater for the definition of rough sets by combining lower and upper approximations and boundaries of other rough sets. Both frameworks take into account that membership of an object into a concept may be unknown.

The fundamental difference between the languages is in the treatment of similarity relations. The first language assumes that similarities between objects are represented by equivalence relations induced from objects with similar descriptions in terms of a given number of attributes. The second language allows the user to define similarity relations suitable for the application in mind and takes into account that similarity between objects may be imprecise. Thus, four-valued similarity relations are used to model indiscernibility between objects, which give rise to rough sets with four-valued approximations, called paraconsistent rough sets. The semantics of both languages borrows ideas and techniques used in paraconsistent logic programming. Therefore, a distinctive feature of our work is that it brings together two major fields, rough sets and paraconsistent logic programming.
I would like to express my gratitude to my supervisor Professor Jan Małuszyński for his guidance, discussions, and comments throughout these years. I would like to express also my appreciation to Professor Andrzej Szalas and Professor Carlos Damásio for their valuable and inspirational collaboration.

I am thankful to my colleagues at VITA and DM for the nice and friendly working environment. A special thanks goes to Katerina Vrotsou for the help with the \LaTeX{} and to Eva Skärblom for helping with all submission process of the thesis.

A final special thank you to my ever loving family and friends who have wholeheartedly supported my wish to conclude this thesis and for being a constant source of strength for me.

Aida Vitória
Norrköping, Sweden
October, 2010
To Silina and Bruno
Contents

1 Introduction 1

1.1 Context of the Work 1

1.2 Rough Sets 2

1.2.1 Rough Sets Overview 2

1.2.2 Knowledge Representation Systems and Rough Sets 3

1.2.3 Boundary Thinning Techniques 9

1.2.4 Applications of Rough Sets in Data Mining 10

1.2.5 Rough Sets and Many-valued Logics 11

1.2.6 Rough Sets and Approximation Spaces 13

1.3 Vagueness and Paraconsistency: A Logic Programming Perspective 15

1.3.1 Paraconsistent Logic Programming 18

1.4 Problem Statement 21

1.5 Contributions 23

1.6 Paper Summaries 25

1.6.1 Summary of Part I 25

1.6.2 Summary of Part II 26

1.6.3 Part I versus Part II 27

1.7 Related Work 28

1.7.1 The System CAKE 28

1.7.2 Fuzzy-rough Sets 28

1.7.3 Rough Description Logics 29

1.7.4 The Language of Inclusion-Exclusion 30

1.7.5 A Rough Sets based Language with a Non-deterministic Three-valued Semantics 31

1.8 Conclusions 31

1.9 Future Work 33

Bibliography 35

2 Publications included in the thesis 45
Complete list of publications


Introduction

1.1 Context of the Work

This thesis presents an approach to knowledge representation combining rough sets and paraconsistent logic programming.

The rough sets framework [Paw91] proposes a method to handle uncertainty due to imprecise or noisy data. In many practical applications, agents have limited information about objects of an universe (e.g. patients) in the sense that only certain attributes (properties) of the objects are known (e.g. blood pressure, temperature). It is then possible that an agent may perceive different objects of the universe as being similar, or indiscernible, considering the available knowledge. However, indiscernible objects may be classified as belonging to different concepts leading to inconsistent information, situation that often results from integration of knowledge from several agents or experts. For instance, one patient might have been diagnosed to have a certain disease while another patient with the same symptoms did not get the same diagnosis. Mohua Banerjee has phrased this idea nicely as follows. “In everyday discourse, we place a ’grid’ over reality. The grid is typically induced by attributes, and then pieces of data having the same values for a set of attributes, cannot be distinguished.” A relevant problem is then how to describe a concept $C$, e.g. patients with a specific disease, using the available knowledge about the objects of the universe. Since agents have a limited capability to discern between different objects, concepts are vague and cannot be described precisely. Rough sets theory has proved to be a suitable technique for managing uncertain and inconsistent knowledge in these cases. The key idea in rough sets theory is that a vague concept $C$, that cannot be described precisely using the existing knowledge, is approximated by means of a pair of precise concepts: a sub-concept describing those objects that definitely belong to $C$ and a super-concept describing those objects that possibly belong $C$. An appealing aspect of rough sets techniques is that they have a clear mathematical foundation.

Knowledge representation systems allow users to define concepts explicitly by examples, as well as create new concepts from existing ones, and reason about the defined concepts. Since inconsistencies frequently occur in knowledge about the real-world, the
problem of representing vague concepts and reasoning in the presence of inconsistency is central to knowledge representation systems. Therefore, rough sets are an interesting framework from the perspective of knowledge representation.

Our work investigates the use of rough sets techniques in knowledge representation systems. More concretely, this thesis focus on systems based on paraconsistent logic programming and tackles the following problems.

- How to define a system that allows users to specify rough sets in terms of other rough sets and reason about them.
- How to allow users to incorporate domain or expert knowledge with the defined rough sets.
- How to extend the basic rough sets formalism so that it can model additional forms of uncertainty, besides contradictory information. First, situations where information about the universe may be incomplete deserves a special attention. Indeed, for some objects there may be no evidence about whether they belong to a certain concept. Second, we should also consider that the knowledge of an agent about similarities between objects of the universe can also be incomplete and inconsistent.

This introduction is organized as follows. In Section 1.2, we review those rough sets notions that influenced our work. Section 1.3 offers a brief overview of logic programming and paraconsistency. The problems addressed in this work are formulated in section 1.4 and section 1.5 describes the main contributions of our work. Section 1.6 summarizes both parts of this thesis. This is followed, in section 1.7, by a comparison of our framework with related work of other authors. Finally, we present our conclusions and discuss future work.

1.2 Rough Sets

This section gives an overview of the rough sets field. The aim of this overview is manifold. Firstly, it gives some background and illustrates the relevance and applicability of rough sets techniques. Secondly, we give the reader a perspective of which aspects of rough sets are influential in our work. Thirdly, it establishes a ground for a comparison between the work reported in this thesis and the work developed by other authors.

1.2.1 Rough Sets Overview

We start by recalling the key ideas underlying the Pawlak rough sets model [Paw91]. This model considers that objects (e.g. cars, patients) are described by attributes (e.g.
color, temperature). Each object of a universe $U$ is described by a vector of attribute values and two objects $x$ and $y$ are indiscernible if they are described by the same vector. Pawlak's indiscernibility relation $R$ is, therefore, an equivalence relation and it induces a partition of $U$ corresponding to the equivalence classes $[x]_R$, for every $x \in U$.

It is not uncommon that the set $A$ of attributes considered is composed of a binary decision attribute $D$ and of a non-empty subset $B$ of conditional attributes, with $A = B \cup \{D\}$. The decision attribute $D$ is usually associated with the concept $C$ ($\neg C$) of objects having value 1 (0) for $D$. Then, the indiscernibility relation $R$ is obtained from those objects having the same values for attributes $B$.

A central problem addressed in the rough sets framework is how to define a concept $C \subseteq U$ in terms of the elementary sets induced by an indiscernibility relation $R$. In the initial Pawlak theory, these elementary sets correspond to the equivalence classes $[x]_R$, for every $x \in U$. If $C$ corresponds to the union of several elementary sets then $C$ is known as a definable set. However in practice, it may not be possible to define precisely $C$ in terms of the partitions obtained from $R$. Thus, $C$ is a rough set (concept) and it is instead characterized by a pair of approximations: the lower approximation, denoted as $C^\downarrow_R$; and, upper approximation, denoted as $C^\uparrow_R$. In the original model, the lower approximation of $C$ is the greatest definable set (w.r.t. set containment $\subseteq$) contained in $C$,

$$C^\downarrow_R = \{x \in U \mid [x]_R \subseteq C\},$$

while the least definable superset of $C$ corresponds to its upper approximation,

$$C^\uparrow_R = \{x \in U \mid [x]_R \cap C \neq \emptyset\}.$$

The region of the universe that is part of the upper approximation but it is not contained in the lower approximation is known as the boundary region, i.e. $C^\uparrow_R \setminus C^\downarrow_R$. Thus, the boundary region corresponds to contradictory information.

Intuitively, the lower and upper approximations of a vague concept $C$ are the set of objects which definitely belong to the concept and the set of objects which possibly belong to $C$, respectively.

The original Pawlak rough sets model has been subject to several extensions making it more suitable for practical applications. Other work has focused on the study of relationships between rough sets models and other frameworks, like modal logics and fuzzy sets, leading to new applications of rough sets theory. Although the author does not aim at presenting here a complete survey of the field, some of the most relevant extensions and connections to other fields are briefly described in the next sections.

1.2.2 Knowledge Representation Systems and Rough Sets

The aim of this section is to give an overview of the connections of rough sets with knowledge representation systems.
Knowledge representation is the field of Artificial Intelligence that focuses on two important problems: representing knowledge symbolically and finding automated methods for reasoning with the represented knowledge. A number of different approaches has been proposed to tackle these problems. Logic programming, systems based on fuzzy sets, and description logics are three of these approaches.

An interesting research direction has been to combine existing knowledge representation systems with rough sets techniques. For example, hybridization of fuzzy and rough sets and integration of rough concepts into description logics has been discussed by several authors. In contrast to these lines of research, the work presented in this thesis establishes a link between rough sets and a specific field of logic programming known as Paraconsistent Logic Programming.

In the following section, we describe briefly a system, named CAKE, that has a purpose similar to the work discussed here, i.e. to represent rough knowledge databases. We then review the major ideas underlying fuzzy-rough sets and integration of rough concepts into description logics. In section 1.7, we compare these systems with the work presented in this thesis.

**System CAKE**

CAKE [DLS02, DLSS06], standing for Computer Aided Knowledge Engineering, is a system to represent knowledge bases of rough concepts. Knowledge of different agents is modelled through a graphical representation that can be viewed as an extension of the entity-relationship diagrams used in relational databases design (see e.g. [AHV95]). This graphical representation is then translated to stratified logic programs, using default negation. Stratification disallows recursive definition of relations (concepts) through default negation (see e.g. [AB94]).

Being a framework founded on the concept of rough sets, CAKE allows the explicit representation of both positive and negative knowledge of an agent and open-world assumption is embedded in the reasoner. The system implicitly associates with each relation (concept) a rough set. However, contradictory information concerning a certain property, corresponding to the boundary region of the denoted rough set, is not distinguished from cases about which there is no knowledge at all about the property. Therefore, the underlying logic of CAKE is three-valued. The truth value UNDEFINED is associated with both boundary (contradictory) cases and with cases for which there is lack of knowledge. Since this aspect may impose practical limitations, CAKE has a mechanism that allows one to resolve inconsistencies by a user-defined voting policy.

An interesting aspect of CAKE is that it allows reasoning through contextually closed queries (CCQ). A CCQ is a tuple \( \langle Q, \Sigma, L, K, I \rangle \), where \( Q \) represents a query (e.g. a first order-formula) to the rough knowledge base \( \Sigma \), \( L \) and \( K \) are disjoint sets of relation symbols, and \( I \) is a set of integrity constraints. The algorithm to answer the query uses then the circumscriptive theory \( \text{CIRC}(I \cup \Sigma, L, K) \) obtained from \( I \cup \Sigma \) assuming that
1.2 Rough Sets

$L$ is the set of predicates whose extensions we want to minimize, while $K$ is the set of predicates whose extensions are fixed during minimization. Thus, answering a C-query implies determining whether

$$\text{CIRC}(I \cup \Sigma, L, K) \models Q,$$

where $\models$ represents logical consequence (e.g. see [BL04]). A discussion about circumscription is behind the scope of this work. The reader is referred to the book [BL04] for more details about circumscription.

CAKE has a fixpoint semantics and the query answering algorithm is co-NPtime complete [DKS04]. The system is expressive enough to represent, for instance, default reasoning and it has been successfully used in a large scale practical application involving UAV (unmanned aerial vehicles) platforms [DLSS06].

Combining Rough Sets and Fuzzy Sets

Theory of fuzzy sets [Zad65] is another successful technique to model uncertain concepts such as "quite tall person" or "warm temperature". Fuzzy sets are a generalization of classical set theory. The central notion in fuzzy sets is that an object of an universe $U$ may belong to a set (concept) $C$ to a certain degree. This is modelled by a membership function $\mu_C : U \rightarrow [0, 1]$ that is a generalization of set characteristic functions. Therefore, a person $a$ may belong to a certain extent to the group of "quite tall persons", e.g. $\mu_C(a) = 0.7$ with $C = "\text{quite tall persons}"$, and simultaneously $a$ is also to some extent seen as a "medium high person", e.g. $\mu_B(a) = 0.2$ with $B = "\text{medium high person}"$. Similar to the usual operations on classical sets, the notions of fuzzy set intersection and union, fuzzy set complement, fuzzy set containment have been defined by several authors, e.g. min-max system proposed by Zadeh [Zad65]. Although different definitions have been proposed for fuzzy sets intersection and union, intersection is usually modelled by a T-norm while union is modelled by a T-conorm (see e.g. [Mar94]).

There is a clear connection between fuzzy sets based systems and many-valued logics [Res69, Got05]. Values of a membership function $\mu_C$ can be interpreted as truth values of a many-valued logic and fuzzy sets operations can be interpreted as logical connectives, e.g. conjunction, disjunction, and negation. Since there are different many-valued logics with their own logical connectives, e.g. Gödel logic (see e.g. [Got10]) and Łukasiewicz logic [Łuk67], it is possible to understand the reason for the existence of several definitions for fuzzy set operators, and consequently, of several fuzzy logical systems.

As rough sets and fuzzy sets are different techniques for modelling uncertainty, two natural questions arise.

- How are these techniques related?
- How to combine them?
The first question has been addressed in [Yao98a]. It is possible to define a (rough) membership function for a rough set $C$. This membership function, presented below, corresponds to the conditional probability $P(x \in C | x \in [x]_R)$, where $[x]_R$ is the equivalence class of an object $x$ and $R$ is the indiscernibility relation modelling indistinguishability between objects of the universe.

$$\mu_C(x) = \frac{|C \cap [x]_R|}{|[x]_R|}$$

Thus, a rough set can be seen as a fuzzy set, such that $\mu_C(x) = \mu_C(y)$, for all objects $y$ indiscernible from an object $x$ (i.e. $y \in [x]_R$). Based on the laws of probability, the membership function above can then be extended to rough sets corresponding $C_1 \cap C_2$, $C_1 \cup C_2$, and $\neg C$. It is worth to note that there is no one-to-one correspondence between rough sets and subsets of $U$, i.e. two different subsets of $U$ may be associated to the same rough membership function. Consequently, the definition of rough membership functions for $C_1 \cap C_2$ and $C_1 \cup C_2$ cannot be only defined in terms of $\mu_{C_1}$ and $\mu_{C_2}$. For instance, $\mu_{C_1 \cup C_2}(x) = \mu_{C_1}(x) + \mu_{C_2}(x) - \mu_{C_1 \cap C_2}(x)$. This leads to the conclusion that rough sets theory corresponds to a class of non-truth-functional fuzzy set systems [Yao98b].

Using fuzzy sets terminology, the lower approximation of a rough set is then the core of $C$, i.e. $\text{core}(\mu_C) = \{x \in U | \mu_C(x) = 1\}$, while the upper approximation is the support of $C$, i.e. $\text{support}(\mu_C) = \{x \in U | \mu_C(x) > 0\}$.

The above view of rough sets as a special type of fuzzy sets also leads to connections of rough sets with a special type of many-valued logics known as probabilistic logics [Nil86, RK03].

As noticed by many authors, rough sets and fuzzy sets model different forms of uncertainty. In rough sets uncertainty originates from indiscernibility, or more generally speaking from similarity, between different objects of the universe under consideration. In the original work presented by Pawlak [Paw91], indiscernibility is modelled by an equivalence relation inducing a partition of the universe into equivalence classes of indiscernible objects. Unlike rough sets theory, fuzzy sets model uncertainty due to imprecise set boundaries, i.e. an object may belong to some extent to a set $C (\mu_C(a) > 0)$ and to its complement $\neg C (\mu_{\neg C}(a) > 0)$. This leads to the second question above: how to combine rough sets and fuzzy sets such that both types of uncertainty are addressed? A first attempt to answer this question was made in [DP92]. Rough-fuzzy sets are the first hybrid of rough sets and fuzzy sets. Rough-fuzzy sets generalize the idea of rough sets to fuzzy sets by defining upper and lower approximations of a fuzzy set considering a crisp indiscernibility relation between the objects of the universe. Rough-fuzzy sets can be generalized to fuzzy-rough sets. In fuzzy-rough sets, fuzzy indiscernibility relations, and consequently fuzzy equivalence classes, are also considered. In both, rough-fuzzy sets and fuzzy-rough sets, upper and lower approximations are fuzzy sets, and therefore, are defined by a membership function.
Given a fuzzy indiscernibility relation $R$, a fuzzy equivalence class $[x]_R$ is defined as $\mu_{[x]_R}(y) = \mu_R(x, y)$, for all $y \in U$. Note that every fuzzy logic has its notion of fuzzy implication, denoted here as $\Rightarrow$, and of conjunction $\land$. The degree of subsumption between two fuzzy sets $A$ and $B$ is usually defined as

$$\text{GLB}_{o \in U}(\mu_A(o) \Rightarrow \mu_B(o)) = \mu_C(o).$$

The lower and upper approximations are therefore fuzzy sets defined below.

$$\mu_{C^+}(x)_R = \text{GLB}_{y \in U}(\mu_{[x]_R}(y) \Rightarrow \mu_C(y)),$$

$$\mu_{C^\land}(x)_R = \text{LUB}_{y \in U}(\mu_{[x]_R}(y) \land \mu_C(y)).$$

These definitions diverge from the crisp definitions of approximations in the sense that the memberships are defined for fuzzy equivalence classes, and not for objects of the universe. Membership functions for objects $\mu_{C^+}(o)$ and $\mu_{C^\land}(o)$, with $o \in U$, can be obtained from the definitions above (for more details see e.g. [JS04]). The interesting aspect to notice here is that the definitions above closely resemble the definitions we propose for approximations of paraconsistent sets, given in section 2.3 of Part II. Informally, paraconsistent sets are characterized by a four-valued membership function that caters for incomplete and contradictory set membership information.

In practice, attributes describing a dataset are often real-valued, e.g. temperature, distance. This poses a problem to the basic rough sets techniques, since they are only suitable for symbolic attributes. For instance, it may be desirable that objects that only differ a few degrees in temperature are considered similar (perhaps, this difference is due to measurement error). One way to cope with the problem is to discretize in advance all real-valued attributes, what may obviously represent a loss of information. For instance, temperature may be discretized as "cold", "warm", and "hot". But, the discretization process does not allow an object to be considered to some extent "warm" and "hot". Although in real-life applications, the boundaries between the sets "cold", "warm", and "hot" may be ill-defined, basic rough sets only allow an object to be either "cold", or "warm", or "hot". Another way to tackle the problem of real-valued attributes is to consider fuzzy sets, i.e. "cold", "warm", and "hot" are modelled as fuzzy sets. If only the decision attribute values are fuzzy then rough-fuzzy sets can be used to build approximations of the fuzzy decision classes. If both decision and (some) conditional attributes are fuzzy then fuzzy-rough sets techniques can be used.

Combining Rough Sets and Description Logics

Rough sets have recently been combined with description logic systems, bringing the rough sets framework to a completely new field of applicability.
Description logics (DLs) [BCM+03] are a specific knowledge representation formalism underlying most of the existing ontology languages. Ontologies represent the vocabulary of some domain, concepts and relationships between the concepts. One of the most successful applications of DLs is the Semantic Web. For example, the description logic SROIQ(D) is the underlying language of the web ontology language OWL 2 [CGHM+08].

Informally, DLs languages allow the representation of concepts (e.g. Student, Male), which denote sets of objects of a universe, and roles (e.g. hasChild), which denote binary relations between objects of the universe. In addition, each language provides a number of operators to build more complex concepts and roles from the primitive ones. Typical operators are concept intersection (\(\sqcap\)), concept union (\(\sqcup\)), negation of a concept (\(\neg\)), quantification, and numeric restrictions. For instance, Student \(\sqcap\) \(\exists\) hasChild.Male is a non-primitive concept representing the set of students with a male child, while Student \(\sqcap\) \(\forall\) hasChild.Male represents the set of students whose children (if any) are only males. Description logic based systems not only have a well-defined model-theoretic semantics, they also offer a number of reasoning services. Checking satisfiability of a concept \(C\), i.e. whether \(C\) can denote a non-empty set of objects, is an example of such reasoning services.

Description logics cannot represent and reason with vague knowledge. Moreover, inconsistencies are dealt in the classical way, i.e. anything can be deduced from an inconsistent knowledge base. Therefore, it has recently been investigated [JWTX09] the possibility of combining rough sets and description logics (a first attempt in this direction was earlier reported in [Lia96]). An extension of the description logic ALC catering for approximate concepts based on rough sets, representation of approximate concept ontologies, and reasoning with approximate concepts is discussed in [JWTX09]. For instance, if Tall represents the vague concept of tall persons then it is possible to represent the set of individuals who certainly are tall, \(\text{Tall}^{+}_{R}\), and those who possibly are tall, \(\text{Tall}^{0}_{R}\), where \(R\) is a similarity relation between persons. It is also possible to state that basketball players are certainly tall, BasketPlayer \(\sqsubseteq\) \(\text{Tall}^{+}_{R}\), or express the concept of individuals who possibly have a male child \(\exists\) hasChild.Male \(\sqsubseteq\) \(\text{Tall}^{0}_{R}\). The most interesting result of this work is that the extended ALC language, i.e. \(ALC^{A}\) with concept approximations, can be be translated to ALC. The key idea is that the lower approximation of a concept can be transformed into an ALC universally quantified concept,

\[
\text{trans}(\text{Tall}^{+}_{R}) = \forall R.\text{trans}(C),
\]

while the upper approximation of a concept can be transformed into an ALC existentially quantified concept

\[
\text{trans}(\text{Tall}^{0}_{R}) = \exists R.\text{trans}(C).
\]

An important consequence of this result is that reasoning services in the extended language can be reduced to standard reasoning services in ALC. For instance, a concept
1.2 Rough Sets

C in the extended ALC language is satisfiable if and only if \( \text{trans}(C) \) is satisfiable in ALC. Therefore, existing systems based on DLs can be readily used for reasoning with rough concepts.

More recently, Bobillo and Straccia studied the possibility of incorporating fuzzy rough sets into DLs [BS09]. They use the same idea as proposed in [JWTX09] and upper and lower approximations become fuzzy DL concepts. They have also implemented two systems for reasoning with fuzzy DLs, FUZZYDL [BS08] and DeLorean [BDGR08].

1.2.3 Boundary Thinning Techniques

Building the lower approximation of a set \( C \) from those elementary sets included in \( C \) can be a too strong requirement. In many practical applications there might be few (or no) elementary sets fully included in \( C \). This leads to large boundary regions. Obviously, this aspect may reduce substantially the predictive capability of a model based on rough sets techniques. Therefore, extensions of basic rough sets techniques that lead to boundary thinning have been proposed by several authors. The framework discussed in the first part of this thesis is expressive enough to allow encoding of two important methods for boundary region thinning: variable precision rough sets model [Zia93, KZ94] and hierarchy structured decision tables [Zia02]. In section 5.1 of Part I, we show several examples of how these methods can be expressed in the language presented. We describe briefly these two methods.

The variable precision rough sets model (VPRSM) [Zia93, KZ94] generalizes the original model by allowing a degree of error in the lower approximations of a concept \( C \) and of its complement \( \neg C \), controlled respectively by two parameters \( \alpha \) and \( \beta \), with \( 0 \leq \beta < P(C) < \alpha \leq 1 \). The intuitive idea is that if the degree of overlapping between an equivalence class \([x]_R\) and \( C \) (\( \neg C \)) is at least (at most) \( \alpha \) (\( \beta \)) then one includes the class in the lower approximation of \( C \) (\( \neg C \)). The lower approximation of \( C \) is then re-defined as

\[
C^+_R = \{ x \in U \mid P(x \in C \mid x \in [x]_R) \geq \alpha \},
\]

and the lower approximation of \( \neg C \) becomes

\[
\neg C^+_R = \{ x \in U \mid P(x \in C \mid x \in [x]_R) \leq \beta \},
\]

where \( P(x \in C \mid x \in [x]_R) = \frac{|C \cap [x]_R|}{|[x]_R|} \).

The VPRSM is a parametric technique for which there is no systematic method to determine the parameters, \( \alpha \) and \( \beta \), best values. To address this particular problem other probabilistic extensions have been considered such as the Bayesian rough sets model [SZ05] and the decision theoretic rough sets model [Yao07]. Unlike the former technique that is characterized by the absence of parameters the later proposes a method to compute the parameters values based on more practical notion of costs.
Another method, described in [Zia02], for boundary region thinning is to associate the boundary region with a new layer of equivalence classes, representing a new finer partition of the boundary. This layer can be obtained in different ways. For example, by associating a new set of attributes with the objects in the boundary, aiming at the new set of attributes induces a finer indiscernibility relation. Alternatively, one can provide more cut points to those numeric attributes that were previously discretized and repeat the discretization process for boundary objects only. Obviously, this process can be applied to the boundary region of the new layer, until the boundary is totally eliminated or is simply small enough, resulting in an hierarchy of partitions.

The idea described above can be concretized in two ways. First, one can associate a new indiscernibility relation with each elementary set in the boundary. Thus, a hierarchical tree structure of partitions is produced. The second method associates the entire boundary region with a new indiscernibility relation producing, consequently, a hierarchical linear structure of partitions.

1.2.4 Applications of Rough Sets in Data Mining

Data mining is, perhaps, the most successful field of application of rough sets techniques. Data mining based rough sets techniques are most suitable for data presented in tabular form, where columns represent attributes describing properties of objects and each row corresponds to a vector of attribute values of an object. Moreover, objects are usually classified as belonging to some decision class $D$ ($\neg D$). Reduct is one of the rough sets notions most useful from data mining point of view. Intuitively, a reduct is a minimal subset of the original attributes that preserves the approximation space and, consequently, concept approximations.

There are numerous areas of interesting applications. For instance, medicine, economics and business, environment, engineering (e.g. control, signal and image analysis), social sciences, molecular biology, and chemistry. In Part I, we illustrate the applicability of our framework with a data mining concrete example.

We do not aim to give here a detailed and complete account of the connections between rough sets and data mining [Sko01]. But, we refer briefly to some of them below.

- **Rough Sets and Data Preprocessing**
  - Handling missing data [GB91].
  - Feature extraction and feature selection [SN99].
  - Combining Principal Component Analysis (PCA) and rough sets techniques for feature selection [Swi01, SS03].

- **Rough sets and Supervised Learning**
  Learning descriptions of the concepts corresponding to the decision classes is one
of the main data mining tasks. These descriptions are typically in the form of if-then classification rules. Consider a set of attributes \( A \) and \( B \) such that \( B \subset A \).

For example, an interesting problem is to find a description of the subset of objects \( O \subseteq U \) for which lack of knowledge about the attributes in \( B \) would worsen classification capability. Note that removing the attributes in \( B \) might imply that some objects “migrate” from the lower approximation of the decision class \( D (\neg D) \) to the boundary region. In practical terms, being able to characterize the objects in \( O \) implies that acquiring the values for attributes in \( B \) only needs to be done for some of the objects not yet classified, and not for the entire universe, without degrading classification capability. If attributes \( B \) correspond in some way to expensive tests then this method leads to savings. The problem just described has been studied in the context of a medical application [KØ99]. We illustrate how this problem can be naturally formulated with the techniques we propose in Part I (sections 5 and 6).

- **Rough Sets and Clustering**
  - Combining K-means with rough sets based techniques [Pet06].
  - Hierarchical agglomerative clustering algorithm using rough sets techniques [SK04, PKBS07]
  - Clustering algorithms integrating techniques from fuzzy-rough sets [MP08, MPB10].

- **Rough Sets and Association Rules**
  Several authors have proposed algorithms for association rule generation based on rough sets techniques, see e.g. [SN99, BGL05].

### 1.2.5 Rough Sets and Many-valued Logics

The rough sets framework discussed in this thesis draws a link between rough sets and many-valued logics. This topic has been also discussed by other authors and we review some of the work done in this direction.

Connections between rough sets and many-valued logics can be established through fuzzy logic and specific algebraic systems. In the former case, as referred in section 1.2.2, the connection to many-valued logics is established through the connection between rough sets and fuzzy sets and the well-known relationship between fuzzy logic and many-valued logics [Got10]. In the latter case, the connection is established by using results that link certain representations of rough sets to particular algebras [Dün97, Pag97], e.g. regular double Stone algebras, and the connection of these algebras with many-valued logics, e.g. correspondence of regular double Stone algebras and three-valued Łukasiewicz logic. More details about these connections are outside
the scope of this thesis. We are instead more interested in connections that have been
directly established between rough sets and many-valued logics, in particular three-valued
logics (see e.g. [MT99]).

Several authors dedicated a special attention to characterize rough sets in terms of
three-valued logics. Usually, this characterization relies on the following key idea. Ev-
ery rough set \( C \) divides the universe into three regions: the positive region corre-
sponding to \( C_R^+ \); the negative region corresponding to all objects in the complement
of the upper approximation of \( C \), i.e. \( U \setminus C_R^+ = (\neg C)_R^+ \); and, the boundary region corre-
sponding to \( C_R^+ \setminus C_R^+ \). Obviously, this approach excludes the possibility of total absence of infor-
mation concerning membership of an object of the universe in concept \( C \). Then, the
truth value true (\( t \)) is associated to those objects in the positive region, while false (\( f \)) is
associated with objects in the negative region. For those objects in the boundary region,
corresponding to contradictory information, a third logical value (\( i \)) is associated with
them.

These ideas are followed in [AK08], where formulas in the proposed logic have the
form \( Cx \), with \( C \) denoting a rough set and \( x \) representing an object of the universe. One
of truth values \( t \), \( f \), or \( i \) is assigned to \( Cx \), if \( x \) belongs to the positive, negative, or
boundary region of \( C \), respectively. Four operations \( (C_1 \cup C_2)x \), \( (C_1 \cap C_2)x \), \( (C_1 \Rightarrow C_2)x \), and \( (\neg C)x \) are part of this logic, where the implication \( C_1 \Rightarrow C_2 \) def \( = \neg C_1 \cup C_2 \).
The following well-known properties of approximations (valid for any binary relation
\( R \))

\[
(C_1 \cup C_2)^+_R \supseteq C_1^+_R \cup C_2^+_R \quad \text{and} \quad (C_1 \cap C_2)^\oplus_R \subseteq C_1^\oplus_R \cap C_2^\oplus_R,
\]

imply that the semantics of the language is not compositional and consequently, the
authors propose a non-deterministic semantics. If \( C_1x \) and \( C_2x \) are both evaluated to \( i \)
then \( (C_1 \cup C_2)x \) is evaluated to one of the truth values in the set \( \{i, t\} \), while \( (C_1 \cap C_2)x \)
is evaluated to a truth value in \( \{f, i\} \). Thus, \( (C_1 \Rightarrow C_2)x \) is evaluated to a truth value in
\( \{f, i, t\} \) and can be represented by the following non-deterministic matrix (similar matrices
could be defined for \( \cup \) and \( \cap \)).

<table>
<thead>
<tr>
<th>( \Rightarrow )</th>
<th>( f )</th>
<th>( i )</th>
<th>( t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f )</td>
<td>( t )</td>
<td>( t )</td>
<td>( t )</td>
</tr>
<tr>
<td>( i )</td>
<td>( i )</td>
<td>( {i, t} )</td>
<td>( t )</td>
</tr>
<tr>
<td>( t )</td>
<td>( f )</td>
<td>( i )</td>
<td>( t )</td>
</tr>
</tbody>
</table>

An interesting result is that the logic introduced in [AK08] can be seen as a gener-
eralization of two well-known three-valued logics, Kleene logic [Kle50] and Łukasiewicz
logic [Łuk67]. If one chooses to evaluate \( i \Rightarrow i \) to \( i \) then the language has a semantics

\footnote{In [AK08], the truth value \( u \) is used instead, but with the meaning we described for \( i \). Therefore for simplicity, we use here \( i \) instead of \( u \).}
based on Kleene logic. Otherwise, if one chooses to evaluate $i \Rightarrow t$ then the language has a semantics based on Łukasiewicz logic. A sequent calculus, without tautologies, is also discussed in [AK08] and it can be used as a sound deduction formalism for the non-deterministic logic and for both “determinizations”, Kleene logic and Łukasiewicz logic. The sequent calculus is also complete in the former case. Completeness for Kleene and Łukasiewicz logics is obtained by adding one specific sequent rule for each logic.

Another language to specify rough sets, having a semantics based on the three-valued Kleene logic, is presented in [MT99]. We discuss this work in section 1.7, where we compare our work with the work reported in [MT99] and [AK08].

1.2 Rough Sets

The initial work on rough sets considers an equivalence relation (i.e. a reflexive, symmetric, and transitive relation) to model indiscernibility. Other authors have investigated extensions of Pawlak ideas by considering other types of binary relations to model more general notions of indiscernibility, or similarity, between objects. Although the work described in Part I assumes that indiscernibility is an equivalence relation, we dropped this restriction in Part II.

Consider a non empty universe $U$ of objects and a binary relation $R \subseteq U^2$ such that $(x, y) \in R$, if $x$ is considered similar to $y$. The set of objects similar to $x$, or neighborhood of $x$, is denoted by $R(x) = \{ y \mid R(x, y) \}$. Let $U/R = \{ R(x) \mid x \in U \}$. The empty set ($\emptyset$) and each member of $U/R$, usually called elementary sets, are seen in the rough sets framework as the building blocks of knowledge about the universe under consideration. This idea is formalized as an approximation space $\langle U, R \rangle$. If $R$ is a reflexive binary relation then the elementary sets of $U/R$ form a covering of $U$. In the particular case that $R$ is an equivalence relation, the covering induced by $R$ is also a partition of the universe such that $R(x) = \{ x \} _R$, for all $x \in U$.

Generalizations of the approximation operators considering an approximation space $\langle U, R \rangle$, where $R$ is a binary relation other than an equivalence relation, are discussed in [SV97, YWL97, Yao98b, SV00]. In [YWL97, Yao98b], the approximation operators are defined as follows

$$C^+ _R = \{ x \in U \mid R(x) \subseteq C \} ,$$

$$C^0 _R = \{ x \in U \mid R(x) \cap C \neq \emptyset \} ,$$

while [SV97, SV00] defined upper approximation as $C^+ _R = \bigcup _{C \subseteq U} R^{-1} (x)$. Although both definitions of upper approximation are equivalent, the latter may be more useful from a computational point of view because it only considers the elements of concept $C$, in contrast to the former that requires the computation of $R(x)$ for all objects $x$ of the universe.
The framework discussed in Part II only imposes that similarities between objects are reflexive relations. Moreover in section 2.5, we prove that our definitions of approximations for paraconsistent sets are equivalent to the approximation operators presented in [SV00], when the paraconsistent sets are the usual two-valued sets.

Several types of relations are explicitly considered in [YWL97]:

- serial relations $R$, i.e. there is at least an $y \in U$ such that $(x, y) \in R$, for all $x \in U$;
- reflexive relations $R$, i.e. $R(x, x)$ for all $x \in U$;
- symmetric relations $R$, i.e. if $R(x, y)$ holds then $R(y, x)$ must also hold;
- transitive relations $R$, i.e. if $R(x, y)$ and $R(y, z)$ hold then $R(x, z)$ must also hold; and
- Euclidian relations $R$, i.e. if $R(x, y)$ and $R(x, z)$ hold then $R(y, z)$ must also hold.

Reflexive relations are also named in the literature as similarity relations, while reflexive and symmetric relations are usually known as tolerance relations.

Properties of approximations have been also investigated, for each of the binary relations above. For instance, serial relations lead to lower approximations that are included in the corresponding upper approximations, i.e. $C^-_R \subseteq C^+_R$, while it is required that $R$ is reflexive for having $C^-_R \subseteq C \subseteq C^+_R$. Moreover, the well-known property of Pawlak rough sets stating that $\emptyset_R = \emptyset$ ($U^+_R = U$) is only valid if relation $R$ is serial. Therefore, the properties of the underlying binary relation determine different rough sets models [Yao98b].

We also investigate, in section 3 of Part II, properties of the proposed approximation operators for paraconsistent sets and contrast them with the properties of lower and upper approximations of usual sets, when similarity (i.e. reflexive) relations are considered.

From a practical point of view, the original approximations spaces $(U, R)$, considered in Pawlak’s work (where $R$ is an equivalence relation), may be too restrictive. For instance, if some attributes are real valued (e.g. temperature) then it may be useful to discard small differences in the values of these attributes. A framework for defining similarity relations that addresses the idea of insignificant differences in attributes’ values of two objects is presented in [SV97]. Therefore, relaxing the conditions imposed on $R$ can also be seen as an alternative to fuzzy-rough sets techniques for the treatment of quantitative attributes. The language proposed in Part II allows the user to define similarity relations suitable for her application. Thus, it is possible to encode in our language a similarity relation that ignores to some extent differences in attributes values of objects.
1.3 Vagueness and Paraconsistency: A Logic Programming Perspective

Investigation of different approximation spaces in connection with rough sets leads naturally to another important research area, the relationship between different rough sets models and modal logics [YL96, Yao96, YWL97]. A discussion about this interesting topic is however outside the scope of this thesis.

1.3 Vagueness and Paraconsistency: A Logic Programming Perspective

Logic programming has been widely recognized as an adequate technique for knowledge representation [BG94, BL04]. A central theme in this thesis is to define a logic programming based language that caters for the specification of concepts denoting rough sets and reasoning with those concepts. Since concepts represented by rough sets are inherently contradictory, we are naturally lead to work with the branch of logic programming that allows representation of (explicit) negation and can reason in the presence of contradictory information. This important field of logic programming is known as Paraconsistent Logic Programming. Consequently, we devote this section to survey informally some of the important notions of paraconsistent logic programming. A more technical discussion of the subject is presented in section 3 of Part I.

The basic way to represent knowledge in logic programming is through sets of definite clauses, known as positive logic programs [Llo87]. A definite clause represents intuitively a universally quantified implication. For instance, the clause

\[\text{mammal}(X) :- \text{cat}(X).\]

is understood as \(\forall X(\text{cat}(X) \Rightarrow \text{mammal}(X))\), stating that if \(X\) is a cat then \(X\) is also a mammal. The declarative semantics of a positive logic program (see e.g. [NM95]) is based on the notion of interpretation. An interpretation \(I\) is expressed as a set of ground atomic formulas of the language (atoms) representing those facts that are true. Note that ground atomic formulas, such as \(\text{cat}(\text{oliver})\), have no variables (e.g. \(X\)). Any ground atom not belonging to \(I\) is false. Thus, an interpretation stipulates which atoms are true and which ones are false. Those interpretations that make every implication represented by a definite clause of the program true are then models of the program. Positive logic programs have always a least model with respect to set inclusion. The least model of a positive logic program defines its meaning. This model is usually computed as the least fixpoint of the immediate consequence operator \(T_P\), for a definite logic program \(P\) [EK76].

Definite clauses disallow the use of negation. Therefore, it is not possible to represent the knowledge below through definite clauses.

- If no train is approaching then cross the rails.
If there is no bus to the city center at a time $H$ then Anna stays at home at time $H$.

This shows that logic programs can only be a valid knowledge representation technique, if there is a mechanism for expressing the falsity of propositions and reason with those propositions.

Two types of negation have been widely discussed for logic programs: default negation [Rei78, AB94], represented as $\text{not}^2$, and explicit negation [GL90, DP98], represented by the symbol $\neg$. Default negation is associated with the closed world assumption, commonly used in deductive databases. With the closed world assumption, the information we have about the world is supposed to be complete. Therefore, any information not contained in the database is false. A typical example is a timetable. We usually assume that if a certain time is not listed in the schedule of a bus then there is no bus departure at that time. The main idea underlying default negation is that anything is false unless it has been stated in the knowledge base to be true. Datalog [AHV95] is a well-known logic programming language for representing deductive databases and queries, allowing the use of default negation. The statement “If there is no bus to the city center at a time $H$ then Anna stays at home at time $H$” can be represented in Datalog as

$$\text{home}(\text{anna}, H) :- \text{not} \text{bus-departure}(H, \text{center}).$$

where $\text{home}(\text{anna}, H)$ represents the piece of information “Anna is at home at time $H$” and $\text{bus-departure}(H, \text{center})$ represents “bus departures at time $H$ to city center”. However, default negation may not be sufficient. For instance, the statement “If no train is approaching then cross the rails.” cannot be represented in Datalog. Notice that default negation is not adequate for expressing this statement, since

$$\text{cross} :- \text{not} \text{train}.$$  

would allow us to conclude that we can cross, when there is no information whether the train is approaching. Hence, other authors proposed to extend the language of definite clauses with another type of negation known as explicit negation [BS89, GL90, Wag93]. The statement above can then be encoded by the extended clause below.

$$\text{cross} :- \neg \text{train}.$$  

Both types of negation, presented above, are used in this thesis.

Logic programs with default negation, known as normal logic programs raise two important problems. First, default negation leads to non-monotonic reasoning (see e.g. [Mak94]). Consider two knowledge bases, $\Sigma_1$ and $\Sigma_2$ such that $\Sigma_2$ was obtained from $\Sigma_1$ by adding some new knowledge. Informally, non-monotonic reasoning implies that some of the conclusions we can draw from $\Sigma_1$ may not be obtained from

\[2\text{The symbol } \sim \text{ is also often used to denote default negation.}\]
1.3 Vagueness and Paraconsistency: A Logic Programming Perspective

$\Sigma_2$, although knowledge has increased. This implies that the common techniques for computing the least model of a logic program $P$ based on the fixpoint of the $T_P$ operator [VEK76, Llo87] are not directly applicable. The $T_P$ operator may simply have no fixpoint, when $P$ is a normal logic program. Second, a logic program with default negation may even not have a least model, in contrast to positive logic programs. Instead, a logic program with default negation may have several minimal models. A classical example is

$$P = \{ p :- \neg q. \}.$$  

The program above has two minimal models, $M_1 = \{ q \}$ and $M_2 = \{ p \}$. This latter aspect raises the issue of defining suitable semantics for normal logic programs.

The semantics of the languages proposed in our work raise problems similar to the ones we have just described. The justification for this is that non-monotonic rough sets based approximation operators are part of the languages.

The problem of defining a suitable semantics for logic programs with default negation has been tackled by different authors. First, a special class of normal logic programs was identified known as (locally) stratified programs [ABW88, Prz88, Gel89] whose meaning is captured by well-supported models [Fag90]. Every stratified program has a unique well-supported model, i.e. every ground atom has an explanation of why is true that does not depend on the atom itself. The well-supported model can be easily computed by calculating the fixpoint of the $T_{P'}$ operator, for every stratum $P'$ of the program. In the second part of this work, we use a similar idea to stratification, since programs expressed in the language proposed in section 5 of Part II may also have several minimal models.

There are however normal logic programs for which there is no stratification, but seem to have an intuitive meaning. For instance, consider the non-stratifiable program

$$P_1 = \{ p :- \neg q., q :- \neg p. \}.$$  

that has two minimal models $M_1 = \{ q \}$ and $M_2 = \{ p \}$. Both models can be intuitively interpreted as different possible sets of beliefs of an agent, given the knowledge base $P_1$, or as different solutions of a problem. The notion of stable model semantics [GL88] formalizes this idea. The importance of stable model semantics in representing incomplete and vague knowledge is illustrated by program $P_1$. Intuitively, it encodes that either $p$ or $q$ should be true, although we cannot be sure which must be true due to our incomplete knowledge about the world. Note that stratified logic programs have a unique stable model that coincides with its well-supported model, while non-stratified programs may have zero or several stable models. The logic programming language discussed in section 3 of part I uses default negation and its semantics is related to stable model semantics. Well-founded semantics [GRS91] is a three-valued semantics for normal logic programs that addresses the problem of existence and uniqueness of a model for these programs. The truth values considered are then
true (t), false (f), and unknown (u). The well-founded model of a normal logic program is contained in the intersection of its stable models. For instance, the program $P_1 = \{ p \leftarrow \neg q, q \leftarrow \neg p \}$ has one well-founded model where both $p$ and $q$ are assigned the truth value $u$. Moreover, the well-founded model of a normal logic program can be computed by a quadratic time algorithm while computing a stable model is a NP-complete problem.

Let us now focus our attention on explicit negation ($\neg$). Logic programs with explicit negation, called extended logic programs, require reasoners that can detect and reason with contradictory (inconsistent) information. In this way, it is possible to capture another aspect of vague knowledge, i.e. contradictory knowledge. This aspect leads us to a well-known field of logic programming known as Paraconsistent Logic Programming, and we devote the next section to it. In both parts of this thesis, we propose languages that use explicit negation and, consequently, our framework relates directly to paraconsistent logic programming.

To finalize, we mention briefly other relevant work in the logic programming field to represent vague knowledge: relevant logic programming [Bol91]; annotated logic programming [KS92]; probabilistic logic programming [NS92]; fuzzy logic programming [Ebr01]; and possibilistic logic programming [DP04, ACG+08]. A formalism for paraconsistent logic programs general enough to capture probabilistic logic programming, possibilistic logic programming, and fuzzy logic programming is presented in [ADP05].

1.3.1 Paraconsistent Logic Programming

The use of explicit negation in logic programs raises the question of what action to take, if contradictory conclusions are obtained from a program. The explosive approach followed in mathematical logic, i.e. anything can be deduced from a contradiction, is not the most suitable for practical applications. Another way to tackle the problem is the belief revision approach [PR91, DP97, DSTW08]. Updating a knowledge base with a new piece of information may introduce inconsistencies, in an initially consistent knowledge base. Belief revision encompasses techniques to allow new information to be added to the knowledge base by making minimal changes in the knowledge base such that no inconsistency arises. A third approach is the one discussed in this section.

Both forms of negation, explicit ($\neg$) and default negation ($\not$) can be used in extended logic programs. The default negation brings the non-monotonic reasoning mechanism to the realm of paraconsistent logic programs. Several two-valued semantics and many-valued semantics have been proposed for extended logic programs. Paraconsistent Stable Model semantics [PR91, Pea93, SI95] is one of those two-valued semantics for extended logic programs. Paraconsistent stable models are stable models where an atom and its explicit negation can occur simultaneously in it. For more technical details about this semantics, the user is referred to section 3 of Part I. The reason for addressing
specifically in some detail the paraconsistent stable model semantics is that reasoning in the language for defining rough relations, presented in section 4 of Part I, is achieved by translating rough programs into extended logic programs with constraints, where both explicit and default negation are used. Paraconsistent stable model semantics is then used to obtain the two-valued models of the transformed extended logic program. Constraints allow rejection of any unwanted stable models. We introduce also a bijection that maps every obtained stable model into a model of the original rough program. The semantics for rough programs assigns rough relations to every predicate symbol occurring in the rough program. We stress here that the translation of lower approximations occurring in a rough program leads to the use of default negation in the obtained extended logic program.

For some extended logic programs no paraconsistent stable model exists. Rough programs also exist that do not have any model, due to their paraconsistent stable model semantics basis. Semi-stable model semantics tackles the problem of extended programs without stable models [SI95]. A many-valued logic, Sakama and Inoue’s logic IX, underlies this semantics. It is left as future research to investigate whether semi-stable model semantics could be used in our framework.

The language proposed in the second part of this thesis is a paraconsistent language, where rough relations can be used, with a semantics based on a four-valued logic. Many-valued logics have been widely used in the paraconsistent logic programming field and we turn now our attention to semantics based on such logics.

Belnap’s four-valued logic [Bel77] is the “kernel” underlying logic of a large number of semantics proposed for extended logic programs. The truth values in this logic are true (t), false (f), inconsistent (i), and unknown (u). The truth values i and u represent contradictory information and lack of knowledge, respectively. Two orderings in this truth space are defined: knowledge-ordering (≤k) and truth ordering (≤t) presented below.

\[
\begin{align*}
    u &<_k f <_k i, & u &<_k t <_k i, \\
    f &<_t u <<_t t, & f &<_t i <<_t t.
\end{align*}
\]

The usual logical connectives ∨ and ∧ are defined with respect to each ordering, i.e. ∨_k, ∨_t, ∧_k, and ∧_t. They represent the meet and the join in each ordering, respectively. A negation operation ¬ is also defined such that ¬t = f, ¬f = t, ¬u = u, and ¬i = i.

Belnap’s logic is as well the departure point for defining the logic underlying the semantics of the language presented in Part II. However, we use a different truth ordering. This change is motivated, for example, by the fact that in Belnap’s logic i ∨ t u = t. A more intuitive result would be to have that i ∨ t u = i. This point is further discussed in section 2 of Part II.

A logic named $\text{FOUR}$ [DP98], extending the Belnap’s logic described above with an implication connective, has been used by several authors [BS89] for defining the semantics of extended logic programs without default negation. For instance, the seman-
tics of (paraconsistent) logic programs introduced by Blair & Subrahmanian in [BS89] is based on the logic \( \textsc{FOUR} \). An interesting contribution of the work presented in [BS89] is a monotonic fixpoint operator with respect to \( \leq_k \) that computes the least model of every program \( \mathcal{P} \), where the least model captures the intended meaning of \( \mathcal{P} \). As noted in [DP98], it is possible to translate the paraconsistent logic programs presented in [BS89] to extended logic programs without default negation, and vice-versa. Thus, the fixpoint operator introduced by Blair & Subrahmanian can be applied in the computation of the least \( \textsc{FOUR} \) model of an extended logic program without default negation.

The major differences between \( \textsc{FOUR} \) and the logic presented in section 2 of Part II of this thesis lie in the truth ordering \( \leq_t \) and the definition of the implication connective giving meaning to the clauses of the language, labelled in our framework as \( \rightarrow_k \). Moreover, in contrast to \( \textsc{FOUR} \), our four-valued logic is equipped with an extra implication connective, denoted \( \rightarrow_t \), that is used for determining the truth value of the literals involving rough sets approximation operators. We have as well defined a fixpoint operator, discussed in section 4 of Part II, used in the computation of the models that capture the meaning of our programs.

Fitting [Fit91a] dedicated special attention to the problem of defining fixpoint semantics of logic programs for which the space of truth values considered forms a bilattice [Gin88]. Fitting’s programs only allow explicit negation. Thus, the non-monotonic operator (with respect to knowledge ordering) default negation is excluded. For a clause \( \mathcal{H} : \neg \mathcal{B} \). of a Fitting’s program, \( \mathcal{B} \) can be a first-order formula, built up from other atomic formulas and using the connectives \( \forall, \exists, \vee_k, \wedge_k, \vee_t, \wedge_t, \) and \( \neg \). A bilattice is a many-valued logic \( \mathcal{R} \) equipped with two orderings, a knowledge ordering \( \leq_k \) and a truth ordering \( \leq_t \), such that \( \langle \mathcal{R}, \leq_k, \wedge_k, \vee_k \rangle \) and \( \langle \mathcal{R}, \leq_t, \wedge_t, \vee_t \rangle \) form complete lattices. Moreover, meet \( \wedge_t \) and join \( \vee_t \) are monotonic with respect to knowledge ordering, and vice-versa. These logics cater for contradictory and missing information. The simplest example of a bilattice is Belnap’s logic. Kifer & Subrahmanian [KS92] have shown that Fitting’s programs can be translated to annotated logic programs providing in this way a model theory for Fitting’s bilattice-based logic programming framework. In contrast to the logics studied by Fitting [Fit91a, Fit91b], the four-valued logic presented in Part II does not form a bilattice because \( \wedge_t \) and \( \vee_t \) are not monotonic with respect to the knowledge ordering. Another more general reason for the semantics of the language presented in Part II departures from Fitting’s work is that the rough sets approximation operators are not monotonic with respect to the knowledge ordering.

In contrast to the work reported in [BS89, Fit91a, KS92], other authors have focused on the problem of defining suitable semantics of extended logic programs using both types of negation, explicit negation and default negation. Many-valued logics have been used as the underlying logic for most of the proposed semantics for these logic programs. For instance, paraconsistent extensions of the well-founded semantics have been considered in [Sak92, PA92, ADP95, DP95]. The Ginsberg’s logic \( \text{VII} \) [Gin88] is
used in [Sak92], while a nine-valued billatice is used in [ADP95, DP95]. A distinctive feature of the latter work is that default and explicit negation are not seen as unrelated. The coherence principle, stating that if $\neg A$ holds then not $A$ should hold too, is embedded in the semantics presented in [ADP95, DP95]. Another relevant work addressing the same problem is presented in [RF97]. The semantics proposed here is based on a nine-valued Kunen-style semantics [Kun89]. Moreover, the authors of [RF97] also show how to define four-valued stable models for extended programs which correspond to those obtained by Gelfond and Lifschitz’s [GL88].

Let us know summarize the major ideas underlying the connection between our work and paraconsistent logic programming. The languages for rough programs presented in this thesis allow explicit negation and rough sets based approximation operators. Default negation is not allowed. Explicit negation leads obviously to a paraconsistent framework, while rough sets based approximation operators introduce non-monotonicity. Rough programs, in the language presented in section 4 of Part I, are translated into extended logic programs, where both explicit and default negation may occur. Paraconsistent stable model semantics is then used to obtain the two-valued models of the transformed extended logic program. As the name suggests, paraconsistent stable model semantics is an extension of stable model semantics that caters for contradictory information. The language for rough programs described in Part II has a semantics based on a four-valued logic. The semantic problems raised by the non-monotonic approximation operators are tackled by considering a special class of rough programs, building on ideas of stratification.

### 1.4 Problem Statement

Reasoning solely on the basis of crisp concepts can be a serious limitation for tackling real-life problems. Therefore, representation of imperfect statements and reasoning with them is a problem that has attracted many researchers.

Rough sets theory has been acknowledged as a technique to handle vague and inconsistent concepts. This theory is particularly suitable for handling vagueness stemming from the incapability of agents to distinguish between similar objects or scenarios, often leading to inconsistent knowledge. Therefore, rough sets have been combined with other techniques used in knowledge representation systems with the aim to obtain more effective systems in their capability to handle vagueness. In this perspective, two major directions of research have been considered. Hybridization of rough sets techniques with fuzzy sets and fuzzy logics led to the development of rough-fuzzy sets and fuzzy-rough sets [DP92, JS02, JS04, CJHS10]. More recently, another major direction of research involves introducing approximate concepts based on rough sets in description logics [BS09, JWTX09].

In this thesis, we investigate the possibility of combining rough sets and logic pro-
gramming techniques, in contrast to the lines of research mentioned above. Thus, the first problem tackled can be described as follows.

- **Problem 1**
  To define a logic programming language that allows users to specify rough sets in terms of other rough sets and reason about them.

In practical applications, experts may only have incomplete information about a concept $C$. For instance in medicine, it is often the case that a complete description of the set of patients at risk for a given disease is unknown. Instead, it may be known that patients satisfying certain conditions are definitely at risk while another group of patients is usually (possibly) not at risk. Therefore, an important aspect of a proposed language is to be able to encode such knowledge about patients at risk and derive meaningful concept approximations. The second problem can then be formulated as follows.

- **Problem 2**
  To investigate how the proposed languages can be used to incorporate domain and expert knowledge. A question arises of how concept approximations can be derived by taking into account not only explicit sets of examples, provided as decision tables, but also the domain knowledge.

The basic rough sets formalism captures vagueness originating from the fact that the universe may be perceived as a family of sets of similar objects, due to an agent’s limited knowledge. Moreover, it is often the case that similarity between objects is modelled as an equivalence relation induced by objects having the same attribute values. However, vagueness has other sources. Firstly, experts may lack complete knowledge about certain objects and, consequently, may not be able to say whether, for instance, an object’s temperature should be classified as hot or just warm. Thus, it should be possible to specify uncertainty about properties (e.g. attributes values) of an object. Secondly, similarity may be defined in different ways. For instance, an agent may consider that, in its perspective, two organisms are similar if they have been in contact and exchanged genetic material. Therefore, an agent should be able to add to its knowledge base its definition of similarity. Thirdly, similarities between objects may themselves be seen as inconsistent and incomplete. For instance, an agent may consider that two objects are similar while another agent may consider that they are perfectly distinguishable. Thus, integrating knowledge of these two agents leads to inconsistencies in the similarities. Additionally, an agent may have total absence of knowledge about the similarity between two concrete objects. Hence, the neighborhood of an object becomes itself a vague concept. These considerations lead us to the third problem investigated in this thesis.

- **Problem 3**
  To propose a rough sets based formalism that takes into account additional forms
of uncertainty. Firstly, information about the universe may be incomplete in the sense that for some objects there may be no evidence about whether they belong to a certain concept. Secondly, an agent should be able to define its own concept of similarity. Thirdly, an object’s neighborhood may itself be a vague concept.

1.5 Contributions

This thesis is organized in two major parts. Part I is a journal paper [Vit05] based on the licentiate thesis of the author. Part II is a substantially extended version of the journal paper [VMS09].

This work contributes to the definition of two paraconsistent logic programming languages catering for the definition of rough concepts and reasoning about them.

- Part I of the thesis focuses on problems 1 and 2 stated above.
  - As a first step in dealing with problem 1, we defined a language that caters for implicit definitions of rough sets obtained by combining different regions of other rough sets (e.g. lower approximations, upper approximations, and boundaries). For instance, the expression below, called rule, states that the lower approximation of a relation $r_1$ is defined as the intersection of the lower approximation of a relation $r_2$ with the boundary of a relation $r_3$.

$$r_1(X_1, X_2) :- r_2(X_1, X_2), r_3(X_1, X_2).$$

We stress that in this part of our work, we assume the usual Pawlak’s indiscernibility relation. The language also allows defining rough sets in terms of explicit examples, as in most currently available systems. For example, the fact $r_2(a, b)$ expresses that all objects in the equivalence class described by the tuple of values $\langle a, b \rangle$ belong to the lower approximation of the concept denoted by $r_2$.

A declarative semantics for the language is also proposed that associates each relation symbol $r$ with a rough relation.

  - The second step in coping with problem 1 was to propose a query language for retrieving information about the concepts represented through the defined rough sets. For instance, the query $(r_2(X_1, X_2), P)$ requests the description of all objects (or equivalence classes) that belong to the lower approximation of the rough relation denoted by $r_2$, with respect to program $P$.

  - We defined a computational engine for the proposed language. This engine is obtained by a translation of the proposed language to the language of
extended logic programs, under the paraconsistent stable model semantics. We also prove the correctness of the proposed translation with respect to the declarative semantics of the language. In this way we establish a link between two important fields, rough sets theory and paraconsistent logic programming.

– Problem 2 is addressed by discussing several motivating applications. These examples show that several useful techniques, such as default reasoning, and extensions to rough sets reported in the literature, and implemented in an “ad hoc” way, can be naturally expressed in our language. For example, we illustrate with examples how the boundary thinning techniques VPRSM and hierarchy structured decision tables can be encoded in our framework.

– We also investigated an extension of the proposed language with numerical measures. This extension is motivated by the fact that numerical measures are an important aspect in data mining applications.

• Part II addresses problem 3, in addition to problems 1 and 2.

– We start by formalizing the notion of paraconsistent rough set. To this end, we first introduce the notion of paraconsistent set as a set equipped with a four-valued set membership function, where the truth values are t (true), f (false), i (inconsistent), and u (unknown). We then redefine the notion of upper and lower approximations to be applicable to paraconsistent sets when the similarity relation between objects of the universe is modelled by a paraconsistent relation, as well. Thus, neighborhood of an object is itself a paraconsistent set. We also prove that paraconsistent rough sets are an extension of the usual rough sets notion based on two-valued similarity relations.

– We investigate several formal properties of paraconsistent rough sets and compare them with standard rough sets properties reported in the literature.

– We formalize a language to represent paraconsistent rough sets and reason with them, such that lower and upper approximations of paraconsistent rough sets can be used in the definition of paraconsistent sets. For example, the rule

$$p(X) : = \sigma_+^\delta(X).$$

defines a paraconsistent relation $p$ as the lower approximation of relation $\sigma$, considering the user defined similarity relation $\delta$. Thus, an object $X$ belongs to the paraconsistent relation denoted by $p$, if $X$ belongs to the lower approximation of the paraconsistent relation denoted by $\sigma$. Note that the language allows the user to define his own notion of similarity.
A fixpoint semantics for a special class of knowledge bases with a layered architecture is presented.

We show several motivating examples and discuss how the introduced framework allows the user to identify subtle boundary cases.

## 1.6 Paper Summaries

In this section, we give a short summary of each of the two parts that compose this thesis, together with a brief description of each of their sections. Part I focuses on problems 1 and 2, while Part II addresses also problem 3.

### 1.6.1 Summary of Part I

We start by defining a language that caters for implicit definitions of rough sets obtained by combining different regions of other rough sets (e.g. lower approximations, upper approximations, and boundaries). In this way, concept approximations can be derived by taking into account domain knowledge. A declarative semantics is also discussed. We then show that programs in the proposed language, called rough programs, can be compiled to extended logic programs, under the paraconsistent stable model semantics. The equivalence between the declarative semantics of a rough program and the declarative semantics of the compiled program is proved. This transformation provides the computational basis for implementing our ideas. Moreover, a query language for retrieving information about the concepts represented through the defined rough sets is defined. Several motivating applications are discussed. Finally, we investigate an extension of the proposed language with numerical measures such as support, strength, accuracy, and coverage. This extension is motivated by the fact that numerical measures are an important aspect in data mining applications.

Part I is organized as follows.

- Section 1 formulates the problem to be addressed and summarizes the major contributions.
- Section 2 gives an introduction to rough sets and a brief overview of how several main problems are addressed in this framework.
- Section 3 surveys some important notions of logic programming and paraconsistent stable model semantics. These topics help the reader to understand the transformation technique applied to the proposed language.
- Section 4 introduces formally a language that caters for implicit definitions of rough sets in terms of other rough sets. We present the declarative semantics of
the language. We also show a transformation of programs in this language to extended logic programs. Moreover, we prove that this transformation is correct with respect to the declarative semantics of the language. In addition, a query language is also defined and an algorithm to obtain answers to the queries is discussed.

- Section 5 demonstrates the feasibility of our approach on practical applications by formulating in our language several problems, presented in the rough sets literature.
- Section 6 proposes an extension of the language with numerical measures.
- Section 7 concludes Part I, makes a comparison with CAKE [DLS02, DKS04, DLSS06], and points to several problems that deserve further research.

1.6.2 Summary of Part II

We consider firstly sets equipped with a four-valued set membership function, called paraconsistent sets. In this approach, set membership function, set containment and set operations (union and intersection) are four-valued, where truth values are \( t \) (true), \( f \) (false), \( i \) (inconsistent) and \( u \) (unknown). We may then have that either an element belongs to a given set (indicated by the value \( t \)), or it does not belong to the set (indicated by \( f \)), or its membership in the set may be unknown (indicated by \( u \)) due to the lack of knowledge, or inconsistent (indicated by \( i \)), perhaps, due to a contradictory evidence or experts uncertainty. To formalize paraconsistent set containment, we introduce a new implication connective.

In the proposed approach, we take into account the fact that in practical applications our knowledge about similarity between objects of the universe can also be incomplete and inconsistent. Thus, similarity is modelled as a reflexive paraconsistent binary relation. This novel aspect of our work has led us to re-think and extend the usual notions of upper and lower approximations. Since we consider similarity relations as four-valued (paraconsistent) sets, there are cases when we cannot establish with certainty whether the neighborhood of an element is a subset of a given set or whether is disjoint with the set. To tackle this problem, we propose upper and lower approximations that are also paraconsistent sets. Based on these ideas, we have then formalized paraconsistent rough sets. Our extension to rough sets is conservative: when only the standard truth values \( t \) and \( f \) are used, all notions we define reduce to the standard operations on sets and rough sets. However, in contrast to the standard rough sets framework, our approach allows to identify different types of boundary cases and, consequently, different degrees of uncertainty.

The next step is to introduce a language to represent and reason with paraconsistent sets, where lower and upper approximations of paraconsistent rough sets can be used
1.6 Paper Summaries

in the definition of other paraconsistent sets. Since upper and lower approximations are non-monotonic operators, programs in the proposed language may have several minimal models. We then provide a fixpoint semantics for the specific class of programs for which there is a stratification and prove that the obtained interpretation is a minimal model of the program. Finally, we discuss several motivating examples.

Part II is organized as follows.

• Section 1 introduces the problem to be tackled and makes some comparisons with related work.

• Section 2 presents the notions of paraconsistent set and paraconsistent rough set. It also proves that paraconsistent rough sets extend the usual notion of rough sets, when the only logical values considered are \( t \) and \( f \).

• Section 3 investigates properties of paraconsistent sets and their approximations.

• Section 4 introduces a rule language to define paraconsistent sets, resembling a paraconsistent Datalog language. A declarative and fixpoint semantics are then described.

• Section 5 extends the language presented in the previous section by allowing paraconsistent set approximations and comparison literals to appear in the body of the rules. The notion of interpretation presented previously in section 4 is suitably extended. The stratification property of programs is introduced and an algorithm to compute a minimal model for stratified programs is presented. This algorithm computes the least model of every stratum by using the fixpoint operator defined in the previous section. Several interesting examples are also discussed.

• Section 6 finalizes Part II with some conclusions and open problems.

1.6.3 Part I versus Part II

We summarize below some of the main differences between the work presented in each of the two parts of this thesis.

• In Part II, set membership is four-valued, while in Part I we use the usual set characteristic function.

• In Part II, set approximations are four-valued, while Part I uses two-valued set approximations.

• In Part II, indiscernibility between objects is modelled by four-valued reflexive relations explicitly defined by the user, while in Part I indiscernibility is an equivalence relation automatically induced by objects having the same attribute values (original Pawlak notion of indiscernibility).
In Part II, it is possible to represent directly membership of an object with respect to a concept. Thus, we drop the assumption of Part I that concepts are sets of tuples of attribute values.

The languages presented in both parts of this thesis allow the use of explicit negation and rough sets based approximation operators. None of the languages allows the use of default negation. In contrast to the work presented in Part I, the operational semantics of the language presented in Part II is not based in a translation technique creating logic programs using default negation from rough programs. However, non-monotonicity is present in the languages discussed in both parts, through the rough sets based approximation operators.

1.7 Related Work

We provide now a comparison of the work discussed in this thesis with related frameworks investigated by other authors. To simplify the reading, we have divided this section in several parts according to the type of framework used for comparison.

1.7.1 The System CAKE

CAKE is another system [DLS02, DKS04, DLSS06] that allows users to define implicitly rough sets, briefly described in section 1.2.2. A comparison between CAKE and the language we propose in the first part of this thesis is included in the end of section 7 of Part I. We refer the reader to that section for more details.

As a final remark, we stress that CAKE resolves any inconsistencies through a voting algorithm. The voting algorithm is either the default one embedded in the system (that associates UNDEFINED with every contradictory case) or it is defined by the user. In contrast to CAKE, the framework discussed in Part I does not require any voting algorithm to resolve inconsistencies. Due to the paraconsistent semantics, it is possible to reason in the presence of contradiction.

1.7.2 Fuzzy-rough Sets

In Part II, we address the problem of building approximations of vague concepts in a vague approximation space. This problem is also tackled by fuzzy-rough sets, where vague concepts modelled as fuzzy sets are approximated using an approximation space of fuzzy equivalence classes. Upper and lower approximations are then expressed as fuzzy-sets [DP92, JS04]. Note that fuzzy equivalence classes are obtained from a fuzzy binary indiscernibility relation satisfying specific axioms. In contrast to fuzzy-rough
sets, we propose that vague concepts are modelled as paraconsistent sets and the approximation spaces considered are families of paraconsistent sets, i.e. neighborhoods of objects are four-valued sets. Lower and upper approximations of a paraconsistent set are also paraconsistent sets. Note also that our notion of inclusion between paraconsistent sets, i.e. $A \sqsubseteq B$, corresponds closely to the notion of degree of subsumption between two fuzzy sets $A$ and $B$, usually defined as $\text{GL}_{B \subseteq U}(\mu_A(o)) \Rightarrow \mu_B(o)$ (with $\Rightarrow$ being the implication of the fuzzy logic being considered). Therefore, the major difference between both frameworks is that fuzzy-rough sets use (implicitly) a many valued logic, while we stay in the realm of a four-valued (paraconsistent) logic. The advantage of using the four-valued logic proposed in this work is twofold. Firstly, the four logical values have a natural and qualitative interpretation: $t$ ($f$) is used to indicate that objects definitely (do not) belong to a concept; $u$ represents total absence of knowledge about membership of an object in a concept; $i$ expresses that an object may belong to a concept, but some doubt is associated with it due to contradictory evidence. Secondly, we suppress the excessive numeric detail of fuzzy sets that may not be suitable in many applications. The dilemma of excessive precision in describing imprecise concepts by fuzzy sets has been noted earlier by other authors [Wit98]. For instance, shadowed sets are introduced in [Wit98] as a technique to reduce fuzzy-sets to three-valued sets while simplifying processing carried out in reasoning with fuzzy sets and enhancing interpretation of results obtained.

### 1.7.3 Rough Description Logics

This thesis tackles the problem of proposing a language to define vague concepts as rough sets and reason with them. Rough description logics [JWTX09, BS09], adding lower and upper concept approximation operators to description logics, have a similar aim. The later is most suitable for knowledge representation and reasoning within the context of ontologies. Since the languages proposed in this thesis are rule languages, in the logic programming style, we can compare and contrast both frameworks in the same way we can discuss (Horn) rules versus description logics. The major point is that they are two complementary alternatives for knowledge representation each having its strengths. For instance, rules allow neither to quantify existentially any variable in the head of a rule nor universal quantification of variables that occur in the rule’s body. Hence, typical description logic axioms such as $\forall \text{has-child.graduate} \sqsubseteq \text{happy-person}$ and $\forall \text{graduate} \sqsubseteq \exists \text{write.thesis}$ cannot be represented by rules. However, in description logics one cannot express that people who live and work in the same city are home workers [GG03], while this could easily be expressed by a clause of a logic program. Therefore, there have been attempts to combine clauses and description logics, for a survey see e.g. [Mał09]. We also wish to point out that a framework for specifying ontologies with approximate concepts based on ideas from rough sets theory is proposed in [DGŁS03], although this work does not build on description
1.7.4 The Language of Inclusion-Exclusion

In [MT99], a language, called inclusion-exclusion language, also addresses the problem of defining intentionally rough concepts and reasoning about them. Like in Part I, the approximation space considered in that work is a partition of equivalence classes induced by objects having the same attributes values. The language is equipped with two atomic expressions of the form \(\text{in}(A=v)\) and \(\text{ex}(A=v)\) expressing that the concept being defined includes, respectively excludes, all objects of the universe having value \(v\) for attribute \(A\). Moreover, the system allows rules to be specified. For instance,

\[
\text{in}(A_1=v_1) \leftarrow \text{ex}(A_2=v_2) \land \text{in}(A_3=v_3)
\]

demonstrates that if a concept includes the objects having value \(v_3\) for attribute \(A_3\) and excludes those objects having value \(v_2\) for attribute \(A_2\) then it must also include those objects having value \(v_1\) for attribute \(A_1\). Negation (\(\neg\)) and disjunction (\(\lor\)) may also appear in the body of the rules, as shown in the example below.

\[
\text{in}(A_1=v_5) \leftarrow \neg\text{in}(A_2=v_4) \land \neg\text{ex}(A_2=v_4)
\]

Rules like the ones above, i.e. not having disjunction in the rule’s body, could easily be expressed in the language we propose in Part I, as follows. Assume that \(c\) is the concept we want to define.

\[
c(v_1,A_2,A_3):= \neg\neg c(W_1,v_2,W_2), c(Z_1,Z_2,v_3).
c(v_5,A_2,A_3):= \exists(W_1,v_4,W_2).
\]

The main idea underlying the language we propose in the first part is that new rough concepts are defined by combining the different regions of other rough concepts, or they are defined by explicit positive and negative examples. The backbone of the language discussed in [MT99] is the atomic expressions \(\text{in}(A=v)\) and \(\text{ex}(A=v)\). A difference to be pointed is that our language allows to express that an example is possibly positive (negative), while the inclusion-exclusion language only allows facts stating that certain objects are definitely positive (negative) examples. Thus in contrast to the inclusion-exclusion language that only allows facts corresponding to lower approximations, we also allow acts using upper approximations (e.g. \(\exists(a,b)\)).

There is also a major difference in the semantics of both languages that is worth to point. The semantics proposed for the language of Part I is (implicitly) a four-valued semantics, since there may be objects of the universe for which we do not know whether they belong to a concept \(C\), i.e. they neither belong to the upper approximation of \(C\) nor to the upper approximation of \(\neg C\). The declarative semantics discussed in [MT99]
Conclusions

is based on the three-value logic of Kleene. Therefore, all objects of the universe either belong to the lower approximation of $C$, or to the lower approximation of $\neg C$, or to the boundary region of $C$. For instance, consider a universe $U = \{o_1, o_2\}$, a theory $T = \{\exists X (X=v)\}$, and assume that $o_1$ is the only object known to have value $v$ for attribute $A$. Unlike the semantics we propose, the semantics of the language of inclusion-exclusion sets object $o_2$ in the boundary region of the vague concept $C$ defined by theory $T$.

1.7.5 A Rough Sets based Language with a Non-deterministic Three-valued Semantics

More recently, Avron and Konikowska have proposed a language [AK08] to define rough concepts based on a non-deterministic three-valued semantics, as we briefly described in section 1.2.5.

In [AK08], membership of an object of the universe in a set $C$ is three-valued, where $i, t, f$ are the truth values used. The logical value $i$ is associated with all objects in the boundary region of a rough concept and information about the universe of objects is assumed to be complete. In our framework, we reserve the logical value $u$ for those objects for which there is lack of information about whether they belong to a set. In contrast to our work, the language of [AK08] cannot express the concept of those objects that are "possibly dangerous and certainly red", since in the latter the user does not have explicit access to the different regions of a concept and, therefore, cannot combine them to form new concepts. On the other side, our work provides neither the possibility for representing $(A \uplus B)^{R+}$ nor $(A \uplus B)^{R\oplus}$, since we would need disjunction ($\lor_t$) in the body of the rules defined in the language of Part II. The language defined in [AK08] allows representation of such concepts. Moreover, reasoning in [AK08] is defined in terms of a sequent calculus.

1.8 Conclusions

It is not uncommon that, in practical applications, vagueness stems from similarities between different objects of the universe. Objects perceived as similar may have been classified as belonging to different concepts leading to possible inconsistencies. Rough sets theory has been recognized as an attractive technique to represent vague concepts. It also has a clear mathematical foundation.

In this thesis, we propose two languages that allow users to represent vague concepts by combining the different regions of rough sets, i.e. lower approximations, upper approximations, and boundary regions. In contrast to most of the systems discussed in the rough sets literature, our framework is based on a four-valued logic because it assumes that every rough concept $C$ induces a partition of the universe into four regions:
the lower approximation of $C$, the lower approximation of $\neg C$, the boundary of $C$, and the region containing those objects for which there is lack of information about their membership in $C$. The latter case is usually not considered by the existing systems.

The proposed languages capture and integrate in a uniform way vague knowledge with two possible sources: knowledge obtained directly from experimental data and encoded as facts; and, domain or expert knowledge expressed as rules. Other authors have also proposed systems to represent knowledge based on rough sets ideas. For instance, rough sets ideas have been incorporated in description logics and in fuzzy logic systems. In this thesis, we go in a different direction by combining rough sets with paraconsistent logic programming.

The language discussed in the first part of this thesis assumes that indiscernibility is induced by objects having the same attributes values, corresponding to the usual Pawlak’s indiscernibility notion modelled by an equivalence relation. Moreover, lower and upper approximations, as well as boundaries, of a concept may appear in the head of a rule. The computational basis for reasoning with the rough sets defined by rough programs expressed in this language is a program transformation. Rough programs are compiled to extended logic programs whose semantics is captured by paraconsistent stable models. Systems like Smodels [Sim10] or dlv [dlv10] can then be readily used to run extended logic programs. The correctness of the proposed compilation technique has been proved. Several useful techniques and extensions to rough sets, reported in the literature [KØ99, Zia02], and implemented in an “ad hoc” way can be naturally expressed in our language. Another important aspect of the work presented in Part I is the definition of a query language to retrieve information about the defined rough sets and patterns implicit in the data. An extension of the language to quantitative measures has also been explored. Quantitative measures are particularly relevant in data mining applications. However, we have restricted this extension to non-recursive rough programs. We also show that this extension allows to capture the variable precision rough sets model [Zia93].

The second part of the thesis starts by proposing a framework for building approximations of vague concepts represented as paraconsistent sets. Paraconsistent sets extend the usual set notion by allowing membership of an element to be unknown or inconsistent. Indiscernibility between objects is modelled by paraconsistent similarity relations, i.e. reflexive binary paraconsistent relations. Consequently, and unlike the first part, similarities between objects are themselves vague concepts. A paraconsistent rough sets algebra $(2^{U \cup \neg U}, \neg, \preceq, \preceq, \cap, \cup, \lhd, \rhd)$ is proposed, where $\preceq$ and $\preceq$ are respectively paraconsistent set lower and upper approximation operators for a given similarity relation $\preceq$. Operations $\cap$ and $\cup$ correspond to paraconsistent set intersection and union, respectively. Although these operations extend the usual two-valued variants, lower and upper approximations are not dual operators in our framework, i.e. $(\neg C)_\preceq \neq \neg((C^{\preceq})_\preceq)$ and $\neg((C^{\preceq})_\preceq) \neq (\neg C)_\preceq$. Lack of duality between the approximation operators also occurs in
1.9 Future Work

Part II introduces then a four-valued rule language which makes it possible to define new paraconsistent sets from given ones. In particular, one can define similarity relations and use them for defining paraconsistent rough sets. The language can be seen as a simple paraconsistent logic programming language without any kind of default negation. On the other hand, in its extended version, the language admits approximation literals and comparison literals in rule bodies, called extended literals. By using comparison literals, one can explicitly check the truth value of a literal, e.g. whether it is $u$. In this way, one can simulate default negation. Moreover, by using approximations literals in comparison literals, it is possible to refer to different boundary cases. Programs written in the extended language may have several minimal models. Therefore, we consider a special class of rough programs, i.e. those for which there is a stratification. Stratified programs are composed as an hierarchy of sub-programs, each forming a stratum, but recursion through extended literals is forbidden. This resembles stratified normal logic programs. The operational semantics of stratified rough programs is based on a polynomial algorithm to compute approximations of a given concept and on a fixpoint operator.

1.9 Future Work

We list below some possible directions for continuing this work in the future.

- Investigation of semantical aspects of the language of Part II.
  - To provide a declarative semantics for the language presented in Part II that characterizes the computed minimal model. In section 5 of Part II, we present an algorithm for computing a minimal model (see definition 5.6 and proposition 5.1) but this model has not been declaratively characterized.
  - To investigate an operational semantics for the language presented in Part II based on a compilation technique of rough programs into logic programs using both explicit and default negation, following a similar idea from Part I. It would not be unlikely that we would be driven to use a semantics based on a many-valued logic, perhaps like the nine-valued logic used in [RF97], for the compiled programs. What many-valued logic to use is an open problem.

- Investigation of extensions of the language of Part II and integration with the ideas of Part I.
  - To investigate the possibility of defining a language building on ideas underlying the languages discussed in both parts of the thesis. For instance, one could extend the language discussed in Part II such that it would allow lower approximations of a rough relation to appear in the head of a clause.
– To lift the stratification restriction for the rough programs considered in Part II. This is perhaps one of the most challenging possible ways to continue this work, since it would be needed to understand in general what is the meaning of recursively defined rough relations. Consider for instance the situation where the definition of a relation $q$ depends on the similarity relation $\sigma$ and the definition of $\sigma$ itself depends on $q$. What can we say about the meaning of $p_\sigma^+$ or $p_\sigma^\ominus$ in these cases?

– To study the possibility of using alternative notions of neighborhood of an object (see definition 2.6 in Part II). For instance from a practical point of view, it could be interesting to consider that the neighborhood of an object $x$ only takes into account those objects $y$ such that $\sigma(x, y) \neq u$.

– To define a query language for extracting information from the programs proposed in Part II.

• Developing an implementation and applications.

– To implement our ideas and investigate possible practical applications. Our work in Part II mostly focus on the theoretical aspects of four-value rough sets approximations and in the definition of a language using those operators.
Bibliography


BIBLIOGRAPHY


