A LOGIC APPROACH TO GRANULAR COMPUTING

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER SCIENCE
UNIVERSITY OF REGINA

By
Bing Zhou
Regina, Saskatchewan

May, 2008

©Copyright 2008: Bing Zhou
NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.
UNIVERSITY OF REGINA

FACULTY OF GRADUATE STUDIES AND RESEARCH

SUPERVISORY AND EXAMINING COMMITTEE

Bing Zhou, candidate for the degree of Master of Science in Computer Science, has presented a thesis titled, *A Logic Approach to Granular Computing*, in an oral examination held on April 16, 2008. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

**External Examiner:** Dr. Yang Zhao, Department of Mathematics and Statistics

**Supervisor:** Dr. Yiyu Yao, Department of Computer Science

**Committee Member:** Dr. Lawrence V. Saxton, Department of Computer Science

**Committee Member:** Dr. Jingtao Yao, Department of Computer Science

**Chair of Defense:** Dr. Peter Gu, Faculty of Engineering
Granular computing is an emerging field of research that attempts to formalize and explore methods and heuristics for human problem solving with multiple levels of granularity and abstraction. A fundamental issue of granular computing is the construction, representation and utilization of granules and granular structures. Basic granules represent the basic pieces of knowledge. A granular structure reflects the connections between different pieces of knowledge. The main objective of this thesis is to examine a logic approach to address this issue. Following the classical interpretation where a concept consists of a pair containing an extension and an intension, we interpret a granule as a pair containing a set of objects and a logic formula describing the granule. The building blocks of granular structures are basic granules representing elementary concepts or pieces of knowledge. They are treated as atomic formulas of a logic language. Different types of granular structures can be constructed by using logic connectives. Within this logic framework, we show that rough set analysis (RSA) and formal concept analysis (FCA) can be interpreted uniformly by using the proposed logic language. The two theories share high-level similarities, but differ in their choices of definable granules and granular structures. Algorithms and evaluation measures can be designed uniformly for both theories.
Acknowledgements

There are many people I would like to thank. Firstly, I would like to thank my Supervisor, Dr. Yiyu Yao. I could not have imagined having a better advisor and mentor during my studies. Without his knowledge, unique perspectives and encouragement, I would not have finished this thesis.

I wish to thank my internal committee member, Dr. Larry Saxton and Dr. Jingtao Yao, for their attentions to my research work and many valuable comments.

I would also like to thank the Faculty of Graduate Studies and Research, Department of Computer Science, and Dr. Yiyu Yao for financial support and the use of computer facilities provided through NSERC grants that were awarded to him. I also want to thank Dr. Jingtao Yao and Dr. Samira Sadaoui-Mouhoub for supporting me when applying for the Graduate Centennial Merit Scholarship and the John Spencer Middleton & Jack Spencer Gordon Scholarship.

Thank-you to my colleagues, Joseph Herbert, Dr. Yan Zhao, Yaohua Chen and Yan Wen for the valuable discussions and all the lunches.

Finally, I would like to say a big ‘thank-you’ to my husband, my Mother and my Father for everything.
Post Defense Acknowledgements

I would like to thank Dr. Yang Zhao, my external examiner, for her instructive and helpful comments on my thesis. Many thanks also go to Dr. Peter Gu, who chaired the defense of my thesis.
# Contents

Abstract i

Acknowledgements ii

Post Defense Acknowledgements iii

List of Figures vii

List of Tables viii

1 INTRODUCTION 1

1.1 Overview of Granular Computing 2

1.1.1 Granular Structures 2

1.1.2 The Triarchic Model of Granular Computing 3

1.2 Motivation 5

1.3 Summary of Contributions 7

1.3.1 A Logic Language $L$ 7

1.3.2 A Unified Interpretation of Rough Set Analysis and Formal Concept Analysis 8

1.4 Organization of the Thesis 8

2 A LOGIC LANGUAGE $L$ 10

2.1 Syntax and Semantics 11

2.2 Differences between $L$ and Other Decision Logic Languages 13
2.3 Two Sub-languages of $\mathcal{L}$ ........................................... 14
2.4 Interpretation of Granules .................................................... 14
2.5 Interpretation of Granular Structures ................................. 16
2.6 Interpretation of Rules .......................................................... 18

3 THE LANGUAGE $\mathcal{L}$ IN ROUGH SET ANALYSIS ........ 19
3.1 Information Tables .............................................................. 19
3.2 Granules in Rough Set Analysis ......................................... 20
3.3 Interpretation of Low and High Order Rules ...................... 23
  3.3.1 Low Order Rules ......................................................... 24
  3.3.2 High Order Rules ....................................................... 25
  3.3.3 Quantitative Measures of Rules .................................... 25
3.4 Rough Set Approaches for the Interpretation of Rules ....... 26
  3.4.1 Standard Rough Set Analysis ........................................ 27
  3.4.2 Dominance-based Rough Set Analysis ........................... 34
3.5 Granular Structures in Rough Set Analysis ....................... 40

4 THE LANGUAGE $\mathcal{L}$ IN FORMAL CONCEPT ANALYSIS .... 44
4.1 Formal Contexts ............................................................... 44
4.2 Granules in Formal Concept Analysis ............................... 49
4.3 Granular Structures in Formal Concept Analysis ............ 51
4.4 Implications and Dependencies in Formal Concept Analysis . 53
  4.4.1 Implications ............................................................. 54
  4.4.2 Functional and Ordinal Dependencies ............................ 55
5 CONCLUSION AND FUTURE RESEARCH

5.1 Summary .................................................. 59

5.2 Future Research .......................................... 61
List of Figures

1.1 The Triarchic Model of Granular Computing [52] .................. 4

3.1 An Example of the $\cap$-closure Granular Structure ............... 40
3.2 An Example of the $\cup$-closure Granular Structure ................ 41

4.1 Concept Lattice of Table 4.3 ..................................... 52
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Rough Set Approaches for Studying Low and High Order Rules</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>An Information Table</td>
<td>31</td>
</tr>
<tr>
<td>3.3</td>
<td>An Information Table with Preference Relations</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>An Example of a Many-valued Formal Context</td>
<td>46</td>
</tr>
<tr>
<td>4.2</td>
<td>Examples of Scale Contexts</td>
<td>47</td>
</tr>
<tr>
<td>4.3</td>
<td>Derived Single-valued Context</td>
<td>48</td>
</tr>
<tr>
<td>5.1</td>
<td>The Unified Interpretation of RSA and FCA</td>
<td>62</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

The study of information and knowledge processing has a long history in computer science. Two research fields, artificial intelligence [23] and cognitive science [26, 27], study this subject in the abstract, in the brain, and in machines. Some of the salient features of human intelligence and problem solving are the conceptualization of a problem at multiple levels of abstraction, the representation of information and knowledge with different-sized granules, the choice of a suitable level of granularity, and the switching of views and granularity in response to changes in environments. An emerging field of study known as granular computing aims at formalizing and exploring these features [2, 38, 44, 48, 51, 58]. The results from granular computing may shed new light on the study of artificial intelligence and cognitive informatics [34, 49].
1.1 Overview of Granular Computing

Granular computing is a new area of research. Its main purpose is to model, state, and solve real world problems at multiple levels of granularity [2,14,39,51]. The fundamental issues, principles and methodologies of granular computing can be interpreted as observing granular structures from different perspectives [44, 46,48,51].

1.1.1 Granular Structures

A granular structure provides a structured description of a system or an application under consideration. Basic granules are the building blocks for forming a granular structure. These granules represent the basic human observations of a problem in the real world. The connections between different granules form different levels of a granular structure which reflect structured knowledge. A granular structure contains at least three basic components [44,48,51]:

- the internal structure of a granule;
- the collective structure of a family of granules;
- the hierarchical structure of a web of granules.

The internal structure of a granule represents the characterization of the granule. Analyzing the internal structure of a granule helps us to understand why objects are drawn together. Granules in the same level are formed with respect
to a certain level of abstraction and collectively show a certain structure. The collective structures are related to granules in other levels.

Granules in different levels are linked by the order relations, interpreted as "more general than" or "more specific than." Granules can be ordered based on their generalities or sizes. For example, in the set-theoretic setting, the size of a granule can be defined by its cardinality. One can define operations on granules so that a set of smaller granules can be formed into larger granules and larger granules can be decomposed into a set of smaller granules. Granules in a higher level can be decomposed into many smaller granules with more details shown at a lower level, and conversely, granules in a lower level can form more abstract larger granules in a higher level. The connections of different levels form a multi-level hierarchical structure. The graph representation of a granular structure is a lattice-like line diagram.

In summary, granular structures are the results of a structured understanding, interpretation, and representation of a real-world problem. Each granular structure represents a particular point-of-view of the problem with multiple levels of granularity. A complete understanding of the problem requires a series of granular structures which should reflect multiple views with multiple levels [4,51].

1.1.2 The Triarchic Model of Granular Computing

There are three different perspectives of granular computing, namely, the philosophy, the methodology and the computation perspectives. These three perspectives together form a triarchic model [41,46,48,51] as shown in Figure 1.1, first
proposed by Yao [52].

From a philosophy perspective, granular computing is a way of structured thinking that focuses on modeling human perception of the cognitive process in reality. It unifies two complementary philosophical views about the complexity of real-world problems, namely, reductionist thinking and systems thinking. Granular computing stresses the importance of conscious effects in thinking with hierarchical structures.

The methodology perspective focuses on methods and strategies for finding structured solutions. As an effective method of structured problem solving, granular computing promotes systematic approaches and practical strategies that are used effectively by humans for solving real-world problems. An important issue is the exploration of granular structures. This involves three basic tasks: constructing granular structures, working within a particular level of the structure, and switching between levels. The methodology of granular computing is inspired by
human problem solving.

The computation perspective focuses on the implementation of computer-based systems. As a paradigm of structured information processing [2], two related notions, namely, representations and processes [15], need to be mentioned. A representation is a formal description and specification of entities formed in information systems. A process can be interpreted as the computational actions that occur in information processing. For example, information granulation and computation with granules are two basic processes of granular computing.

1.2 Motivation

Granular computing can be considered as an umbrella term covering topics that concern granularity. It has been studied either implicitly or explicitly in many fields [2, 44, 48, 51, 58]. Principles and strategies of granular computing have appeared in many branches of science and fields of computer science. Researchers in many branches of science are describing the same notion using different languages. Granular computing attempts to extract the commonalities from existing fields to establish a set of generally applicable principles, to synthesize their results into an integrated whole, and to connect fragmentary studies in a unified framework. However, the advancement of granular computing suffers from the lack of theoretical foundations that enables us to explain some fundamental roles, such as the construction, representation and utilization of granules and granular structures.
Logic is known as a unifying theoretical framework from which the semantics of data models can be understood [12, 35]. It provides a ready-made set of tools whereby a large variety of data types can be represented and studied. Applying these highly-developed tools to granular computing gives deep insights into the different perspectives of granular computing. This justifies the need for a more complete granular computing theory. The main objective of this thesis is to provide a logic approach to granular computing. Instead of representing knowledge in a data table, we abstract knowledge into a set of granules which can be interpreted by logical tools. In this way, one can directly work on relations between objects and their properties by addressing them with basic granules, while revealing the fundamental differences between various granular computing models.

The origins of the granular computing ideology can be found in the literature of rough set analysis. As a concrete model of granular computing, rough sets have inspired a variety of research in logic. Pawlak [19] proposed a decision logic language to describe the object-attribute relationships in an information table. However, this language has limited expressing powers and it can not be used for other granular computing models. This brings a call for more robust languages for granular computing. In this thesis, we introduce a logic language as an extension of the decision logic language used by Pawlak. Instead of expressing the atomic formulas by a particular concrete type of conditions, we treat them as abstract notions to be interpreted in different applications. This flexibility allows us to apply the language to study a much wider class of granular computing models.
and analyze them under a unified logic framework.

1.3 Summary of Contributions

The work presented in this thesis establishes a logic foundation for granular computing. The main contributions are summarized below.

1.3.1 A Logic Language $\mathcal{L}$

We introduce a logic language $\mathcal{L}$ to study granular computing in a logic setting. The language is interpreted in Tarski's style through the notion of a model and satisfiability $[5, 20, 29, 41, 53]$. The model is defined as a pair consisting of a domain and knowledge about the domain. The meaning of a formula is given by a set of objects that satisfy the formula. Similar to the representation of a concept by a pair of extension and intension $[21, 25, 28, 33]$, a granule is interpreted as a pair of a set of objects of the domain and a formula of the language $\mathcal{L}$. Thus, we can interpret granular computing in both a set-theoretic setting and a logic setting. Basic granules are represented by atomic formulas. An object satisfies a formula if the object has the properties as specified by the formula.

The fundamental elements of granular computing, such as basic granules, family of granules, granular structures, and rules are precisely interpreted by using the language $\mathcal{L}$. The tasks of granular computing are formalized during this process. By examining all these tasks in a unified logic framework and extracting their commonalities, we are one step closer to developing a general theory for
1.3.2 A Unified Interpretation of Rough Set Analysis and Formal Concept Analysis

As an application of this research, we demonstrate the usefulness of the language $\mathcal{L}$ by unifying two existing granular computing models, namely, rough set analysis [17, 19, 20] and formal concept analysis [37, 45]. Rough set analysis studies the object-attribute relationships in an information table. Formal concept analysis studies these relationships in single-valued and many-valued formal contexts. The two theories explore different types of definability of a granule, and, consequently, different granular structures. With the introduced language, the two theories can be interpreted in a unified way. On the one hand, the two theories share high-level similarities in their treatments of granular structures. On the other hand, they use different atomic formulas, definable granules, and granular structures formed by definable granules. The unified study of the two theories not only demonstrates the potential of the logic approach to granular computing, but also brings more insights into data analysis using the two theories.

1.4 Organization of the Thesis

In Chapter 2, we introduce the formulation of the language $\mathcal{L}$, as well as the interpretation of granules, granular structures, and rules. In Chapters 3 and 4, we consider the interpretations of the language $\mathcal{L}$ in both rough set analysis
and formal concept analysis respectively. The construction of granules, granular structures, and different types of rules are discussed in both theories. We conclude our research and suggest the directions of future research in Chapter 5.
Chapter 2

A LOGIC LANGUAGE $\mathcal{L}$

In this chapter, we introduce a logic language $\mathcal{L}$ to facilitate our studies of the properties of the various models of granular computing. Similar to any existing formal languages [13], the language $\mathcal{L}$ consists of two aspects: syntax and semantics. The syntax of $\mathcal{L}$ is defined based on a 0-order, propositional logic consisting of atomic formulas and compound formulas. The semantics of $\mathcal{L}$ are given in Tarski’s style [5, 20, 29, 41, 53], namely, a set-theoretic, two-valued semantics. In such a semantics, every formula of $\mathcal{L}$ is assigned an extension, such that each object is either in that extension or not in that extension. Given any formula and any object, the formulas of $\mathcal{L}$ are not vague, it either applies or fails to apply to that object. Tarski’s style semantics allow us to study granular computing in both a set-theoretic setting and a logic setting, and are more appropriate for us to precisely define granules, granular structures and rules than many-valued semantics [3].
2.1 Syntax and Semantics

The language $\mathcal{L}$ is a 0-order, propositional language. Its syntax can be formally defined based on the standard propositional language. In general, one may also consider a first-order language. For the discussion of this thesis, a propositional language is sufficient. Atomic formulas are the building blocks of the language $\mathcal{L}$, denoted by

$$\mathcal{A} = \{p, q, \ldots\}.$$ 

Each atomic formula may be interpreted as representing one piece of basic knowledge. We assume that they are the elementary units that one uses to represent and understand a real world problem. The physical meaning of atomic formulas becomes clearer in a particular application. In general, an atomic formula corresponds to one particular property of an object under discussion. The construction of atomic formulas is an essential step of knowledge representation. The set of atomic formulas provides a basis on which more complex knowledge can be represented. Compound formulas can be built recursively from atomic formulas by using logic connectives, written as:

(i). $(\neg \phi)$,
(ii). $(\phi \land \psi)$,
(iii). $(\phi \lor \psi)$,
(iv). $(\phi \rightarrow \psi)$,
(v). $(\phi \leftrightarrow \psi)$,
where \( \phi \) and \( \psi \) are formulas of the language \( \mathcal{L} \).

The semantics of the language \( \mathcal{L} \) is given in the Tarski’s style by using the notions of a model and satisfiability. The model is defined as a pair \( \mathcal{M} = (D, K) \), where \( D \) is a nonempty set of objects called the domain of \( \mathcal{L} \), denoted by \( D = \{x, y, \ldots\} \), and \( K \) is available knowledge about objects of \( D \). For example, in the decision logic used in rough set theory, the knowledge \( K \) is a set of finite attributes used to describe the set of objects of \( D \). The satisfiability of an atomic formula by objects of \( D \) is viewed as the basic knowledge describable by the language \( \mathcal{L} \). In general, an object satisfies a formula if the object satisfies the properties as specified by the formula. For an atomic formula \( p \), we assume that the available knowledge \( K \) enables us to decide if an object \( x \in D \) either satisfies \( p \), written as \( x \models p \), or does not satisfy \( p \), written as \( x \not\models p \). Let \( \phi \) and \( \psi \) be two formulas, the satisfiability of compound formulas is defined as follows:

1. \( x \models \neg \phi \Leftrightarrow x \not\models \phi \),
2. \( x \models \phi \land \psi \Leftrightarrow x \models \phi \) and \( x \models \psi \),
3. \( x \models \phi \lor \psi \Leftrightarrow x \models \phi \) or \( x \models \psi \),
4. \( x \models \phi \rightarrow \psi \Leftrightarrow x \models \neg \phi \lor \psi \),
5. \( x \models \phi \leftrightarrow \psi \Leftrightarrow x \models \phi \rightarrow \psi \) and \( x \models \psi \rightarrow \phi \).

To emphasize the roles played by the set of objects \( D \) and the set of atomic formulas \( \mathcal{A} \), we also rewrite the language \( \mathcal{L} \) as \( \mathcal{L}(\mathcal{A}, D, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\}) \).

The model \( \mathcal{M} \) describes the meaning of formulas in \( D \). If formulas are interpreted in the model, then each formula becomes meaningful and describes
properties of some objects [19]. The meaning of the formula is the set of all objects having the properties described by the formula. If \( \phi \) is a formula of \( \mathcal{L} \), the meaning of \( \phi \) in the model \( \mathcal{M} \) is the set of objects defined by:

\[
m(\phi) = \{ x \in D \mid x \models \phi \}.
\] (2.1)

That is, \( m(\phi) \) is the set of objects satisfying the formula \( \phi \). This establishes a correspondence between logic connectives and set-theoretic operators. The following properties hold [19]:

1. \((C1). \quad m(\neg \phi) = \neg m(\phi),\)
2. \((C2). \quad m(\phi \land \psi) = m(\phi) \cap m(\psi),\)
3. \((C3). \quad m(\phi \lor \psi) = m(\phi) \cup m(\psi),\)
4. \((C4). \quad m(\phi \rightarrow \psi) = \neg m(\phi) \cup m(\psi),\)
5. \((C5). \quad m(\phi \leftrightarrow \psi) = (m(\phi) \cap m(\psi)) \cup (\neg m(\phi) \cap \neg m(\psi)),\)

where \( \neg m(\phi) = D - m(\phi) \) denotes the set complement of \( m(\phi) \).

### 2.2 Differences between \( \mathcal{L} \) and Other Decision Logic Languages

A fundamental difference between the language \( \mathcal{L} \) and other decision logic languages is the treatment of the set of atomic formulas. In early works, atomic formulas are defined using specific forms. For example, atomic formulas can be defined in an information table based on the values of attributes [19,53]. In this
thesis, we treat atomic formulas as abstract notions that need to be made concrete in different applications. Many concrete examples of the language can be obtained by various definitions of atomic formulas. The construction of the set of atomic formulas and the model \( \mathcal{M} \) depends on a particular application. For modeling different problems, we may choose different sets of atomic formulas and models. Therefore, when switching from one model to another, the structures of the language \( \mathcal{L} \) remain the same, but the flexibility in the semantic interpretation enables us to describe a variety of problems.

2.3 Two Sub-languages of \( \mathcal{L} \)

The language \( \mathcal{L} \) provides a formal method for describing and interpreting rules in data mining and machine learning [56, 59]. In many situations, one is only interested in certain types of rules. For example, rules that contain only the logical connective \( \wedge \). This requires us to consider a restriction of the language \( \mathcal{L} \) to certain logical connectives. In this thesis, we consider two sub-languages of \( \mathcal{L} \). The first uses only the conjunctive connective \( \wedge \), written as \( \mathcal{L}_\wedge(A, D, \{\wedge\}) \), and the second uses only the disjunctive connective \( \vee \), written as \( \mathcal{L}_\vee(A, D, \{\vee\}) \).

2.4 Interpretation of Granules

In the study of concepts [16, 18, 21, 25, 28, 33], many interpretations have been proposed and examined. The classical view regards a concept as a unit of thought consisting of two parts, namely, the intension and extension of the concept [5,
33, 37, 47]. An intension specifies the necessary and sufficient conditions for an object being a member of a specific set. An extension contains every object that is a member of a specific set. By using the language \( \mathcal{L} \), we obtain a simple and precise representation of a concept in terms of its intension and extension. That is, a concept is defined by a pair \( (m(\phi), \phi) \). The formula \( \phi \) is the description of properties shared by objects in \( m(\phi) \), and \( m(\phi) \) is the set of objects satisfying \( \phi \). A concept is thus described jointly by its intension and extension. This formulation enables us to study concepts in a logic setting in terms of intensions and in a set-theoretic setting in terms of extensions.

Following the classical view of a concept, we also treat a granule as the pair \( (m(\phi), \phi) \): the set of objects \( m(\phi) \subseteq D \) and a logic formula \( \phi \). Thus, we obtain both a set-theoretic description and a logic description of granules. In the remaining thesis, we use the two descriptions interchangeably. However, it should be noted that for the same set of objects \( X \subseteq D \), one may find more than one formula in the language such that \( m(\phi) = X \).

We can classify granules by the number of atomic formulas in their intensions. In the sub-language \( \mathcal{L}_{\land}(\mathcal{A}, D, \{\land\}) \), a granule involving \( k \) atomic formulas is called a \( k \)-conjunction. A \( k \)-conjunction granule is more general than its \( (k + 1) \)-conjunctions, and more specific than its \( (k - 1) \)-conjunctions. In the sub-language \( \mathcal{L}_{\lor}(\mathcal{A}, D, \{\lor\}) \), a granule involving \( k \) atomic formulas is called an \( k \)-disjunction. In this case, an \( k \)-disjunction granule is more general than its \( (k - 1) \)-disjunctions, and more specific than its \( (k + 1) \)-disjunctions.
2.5 Interpretation of Granular Structures

The introduction of the language $\mathcal{L}$ also enables us to study granular structures in logic terms. There are at least two ways to construct a granular structure based on the order relations between granules. As examples, we briefly discuss two granular structures called $\cap$-closure and $\cup$-closure [56, 59].

Let $GS_\cap(\mathcal{L}_\land)$ denote the $\cap$-closure granular structure. We can formally define it by the sub-language $\mathcal{L}_\land(A, D, \{\land\})$, written as:

$$GS_\cap(\mathcal{L}_\land) = (\text{Def}(\mathcal{L}_\land(A, D, \{\land\})), \cap),$$

where $\text{Def}(\mathcal{L}_\land(A, D, \{\land\}))$ is the family of granules defined by the sub-language $\mathcal{L}_\land(A, D, \{\land\})$.

The process of constructing an $\cap$-closure granular structure is a top-down process, which involves dividing a larger granule into smaller and lower level granules. Each granule is labeled by the formulas of the language $\mathcal{L}_\land(A, D, \{\land\})$. At the top-most level, the most general granule is labeled by the formula $\top$, which is satisfied by every object, that is, $m(\top) = D$. The next level are the basic granules labeled by atomic formulas. The intersections of two basic granules produce the next level of granules labeled by the conjunction of the two atomic formulas, and so on. Finally, at the bottom level, we close the structure by the most specific granule labeled by the formula $\bot$, which is not satisfied by any object of the formal context, that is, $m(\bot) = 0$. In the set-theoretic setting, the $\cap$-closure granular structure is in fact a closure system that contains $D$ and is closed under set intersection.

16
Let $GS_u(\mathcal{L}_v)$ denote the $\cup$-closure granular structure. We can formally define it by the sub-language $\mathcal{L}_v(A, D, \{v\})$, written as:

$$GS_u(\mathcal{L}_v) = (\text{Def}(\mathcal{L}_v(A, D, \{v\})), \cup),$$

where $\text{Def}(\mathcal{L}_v(A, D, \{v\}))$ is the family of granules defined by the sub-language $\mathcal{L}_v(A, D, \{v\})$.

The process of constructing an $\cup$-closure granular structure is a bottom-up process, which involves the process of forming a larger and higher level granule with smaller and lower level granules. At the bottom-most level, the most specific granule is labeled by the formula $\bot$, which is not satisfied by any object. The next lowest level consists of the basic granules labeled by atomic formulas. The unions of two basic granules produce the upper level of granules labeled by the disjunction of the two atomic formulas, and so on. Finally, at the top level, we close the structure by the most general granule labeled by the formula $T$.

In the set-theoretic setting, all possible granules form a structure known as the power set of the domain $D$, written as $2^D$. For a concrete granular computing model, one is only interested in a subset of $2^D$ which is considered “meaningful” in that model, called a subsystem $G \subseteq 2^D$. The choice of $G$ relies on the domain knowledge. For example, in rough set analysis [19,20], the granules of $G$ are granules that are definable by a set of attributes, and they are the union of equivalence classes. In formal concept analysis [7,36], granules of $G$ are the extensions of formal concepts. In knowledge spaces theory [6], granules of $G$ are the feasible knowledge statements. With respect to the language $\mathcal{L}$, a granule is definable if we can find a formula whose meaning set is that granule.
2.6 Interpretation of Rules

By identifying the basic pieces of knowledge in different granular computing models as basic granules, the differences and high-level similarities of these models can be abstracted and serve as foundations for further data analysis. Rules, as a commonly used form for representing knowledge, can be studied uniformly for different granular computing models. The conditions in rules can be interpreted as granules while entire rules as the relationships between granules. Different types of granules may generate different kinds of rules. Rule learning algorithms can be designed based on granular structures. Different rule applications, such as decision-making, prediction, classification, and pattern recognition, can be systematically analyzed under this scheme.
Chapter 3

THE LANGUAGE $\mathcal{L}$ IN ROUGH SET ANALYSIS

Rough set analysis [19, 20] studies relationships between objects and their attribute values in an information table. We use rough set analysis as a concrete granular computing model to show the usefulness and the flexibility of the language $\mathcal{L}$.

3.1 Information Tables

An information table provides a convenient way to describe a finite set of objects by a finite set of attributes [19]. Formally, an information table can be expressed as:

$$S = (U, At, \{V_a | a \in At\}, \{\{R_a\} | a \in At\}, \{I_a | a \in At\}),$$
where

\[ U \text{ is a finite nonempty set of objects,} \]
\[ At \text{ is a finite nonempty set of attributes,} \]
\[ V_a \text{ is a nonempty set of values for } a \in At, \]
\[ \{R_a\} \text{ is a family of binary relations on } V_a, \]
\[ I_a : U \rightarrow V_a \text{ is an information function.} \]

Each information function \( I_a \) maps an object in \( U \) to a value of \( V_a \) for an attribute \( a \in At \).

The above definition of an information table considers more knowledge and information about relationships between values of attributes. Each relation \( R_a \) can represent the similarity, dissimilarity, or ordering of values in \( V_a \) [5]. The equality relation = is a special case of \( R_a \). The standard rough set theory uses the trivial equality relation on attribute values [19].

Pawlak and Skowron [20] consider a more generalized notion of an information table. For each attribute \( a \in At \), a relational structure \( \mathcal{R}_a \) over \( V_a \) is introduced. A language can be defined based on the relational structures. A binary relation is a special case of relational structures.

### 3.2 Granules in Rough Set Analysis

The indiscernibility relation is a fundamental notion in rough set theory. It indicates why objects are drawn together to form granules. By using the language
we can formally define an indiscernibility relation in an information table. For a subset \( A_0 \subseteq A \), two objects \( x, y \in U \) are indistinguishable if no formula in \( A_0 \) can distinguish them. They can then be put into the same granule. Let us define a mapping from \( U \) to \( A \) as follows:

\[
m'(x) = \{ p \in A \mid x \models p \}.
\]

That is, \( m'(x) \) is the set of atomic formulas satisfied by \( x \). For a subset \( A_0 \subseteq A \), the indiscernibility relation can be defined by:

\[
x \sim_{A_0} y \text{ iff } m'(x) \cap A_0 = m'(y) \cap A_0.
\]

This definition of the indiscernibility relation is slightly different from the conventional definition [19]. One can easily show that the conventional definition is a special case with a properly chosen \( A_0 \).

Based on the indiscernibility relation, we can construct the language \( \mathcal{L} \) by using an information table as the model \( \mathcal{M} \). There are at least two types of granules that can be formally defined. They represent the two different types of knowledge that one can derive from an information table [57].

First, we consider objects of the domain \( D \) as objects in the universe \( U \). The set of atomic formulas are constructed from an attribute-value pair. With respect to an attribute \( a \in At \) and an attribute value \( v \in V_a \), an atomic formula of the language \( \mathcal{L} \) is denoted by \( (a, R_a, v) \). An object \( x \in U \) satisfies an atomic formula \( (a, R_a, v) \) if the value of \( x \) on attribute \( a \) is \( R_a \)-related to the value \( v \), written as \( I_a(x) R_a v \):

\[
x \models (a, R_a, v) \text{ iff } I_a(x) R_a v.
\]
We rewrite the language as $L(\{(a, R_a, v)\}, U, \{\neg, \land, \lor, \to, \leftrightarrow\})$. The granule corresponding to the atomic formula $(a, R_a, v)$, namely, its meaning set, is defined as:

$$m(a, R_a, v) = \{x \in U \mid I_a(x)R_a v\}.$$ 

Granules corresponding to the atomic formulas are basic granules in an information table. Granules corresponding to the compound formulas can be formally defined according to Equation (2.1).

A subset or a granule $X \subseteq U$ is definable in an information table if and only if there exists a formula $\phi$ in the language $L(\{(a, R_a, v)\}, U, \{\neg, \land, \lor, \to, \leftrightarrow\})$ such that,

$$X = m(\phi).$$

Otherwise, it is undefinable. Accordingly, the family of granules is defined as:

$$\text{Def}(L(\{(a, R_a, v)\}, U, \{\neg, \land, \lor, \to, \leftrightarrow\})) = \{m(\phi) \mid \phi \in L(\{(a, R_a, v)\}, U, \{\neg, \land, \lor, \to, \leftrightarrow\})\}.$$

In this way, the language $L$ only enables us to define certain subsets of $U$. For an arbitrary subset of $U$, we may not be able to construct a formula for it. In other words, depending on the set of atomic formulas, the language $L$ can only describe a restricted family of subsets of $U$.

Second, we consider objects of the domain $D$ as object pairs in $U \times U$. With respect to an attribute $a \in At$, an atomic formula of the language $L$ is denoted by $(a, R_a)$. A pair of objects $(x, y) \in U \times U$ satisfies an atomic formula $(a, R_a)$ if the
value of \( x \) is \( R_a \)-related to the value of \( y \) on the attribute \( a \), that is, \( I_a(x) R_a I_a(y) \).

We show this with the following:

\[
(x, y) \models (a, R_a) \iff I_a(x) R_a I_a(y).
\]

We rewrite the language as \( \mathcal{L}((\{a, R_a\}, U \times U, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\}) \). The granule corresponding to the atomic formula \((a, R_a)\), i.e., the meaning set, is defined as:

\[
m(a, R_a) = \{(x, y) \in U \times U \mid I_a(x) R_a I_a(y)\}.
\]

Accordingly, the granules corresponding to the compound formulas of the language \( \mathcal{L}((\{a, R_a\}, U \times U, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\}) \) can be defined by Equation (2.1), and the family of all definable sets or granules is defined as:

\[
\text{Def}(\mathcal{L}((\{a, R_a\}, U \times U, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\})))
\]

\[=
\{m(\phi) \mid \phi \in \mathcal{L}((\{a, R_a\}, U \times U, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\})\}\}.
\]

For granules that are undefinable, it is impossible to construct a formula with the set as its meaning set. In order to characterize an undefinable set, one may approximate it from below and above by two definable sets, called the lower and upper approximations in rough set theory.

### 3.3 Interpretation of Low and High Order Rules

Rules are a commonly used form for representing knowledge. The language \( \mathcal{L} \) provides a formal method for describing and interpreting conditions in rules as granules and rules as relationships between granules. In Section 3.2, two types
of granules are constructed in an information table. One consists of a set of objects that share the same attribute value. The other consists of object pairs that cannot be distinguished based on the values of an attribute. Two types of rules, called low and high order rules, can be defined to describe relationships between these two types of granules accordingly [57].

A low order rule expresses connections between attribute values of the same object. Classification rules are a typical example of low order rules. For example, a classification rule may state that "if the Hair color is blond and Eye color is blue, then the Class is +." A high order rule expresses connections of different objects in terms of their attribute values. Functional dependencies are a typical example of high order rules. For example, a functional dependency rule may state that "if two persons have the same Hair color, then they will have the same Eye color." In general, a high order rule summarizes the results of a family of low order rules.

We express rules in the form $\phi \Rightarrow \psi$ by using formulas of the language $\mathcal{L}$.

### 3.3.1 Low Order Rules

A low order rule can be derived according to the relationships between the first type of granules. For easy understanding, we express the formula $(a, R_a, v)$ in another form based on the definition of satisfiability of the atomic formulas.

An example of a low order rule is given as:

**Low Order rule:**

$$\bigwedge_{i=1}^{n} (I_{a_i}(x) \ R_{a_i} \ v_{a_i}) \Rightarrow \bigwedge_{j=1}^{m} (I_{d_j}(x) \ R_{d_j} \ v_{d_j}),$$
where \( x \in U, v_{a_i} \in V_{a_i}, v_{d_j} \in V_{d_j}, a_i \in At, \) and \( d_j \in At. \) For simplicity, we only use conjunction \( \land \) in the rule.

### 3.3.2 High Order Rules

A high order rule can be derived according to the relationships between the second type of granules. Similarly, we express the formula \((a, R_a)\) in the form based on the definition of satisfiability of the atomic formulas.

An example of a high order rule is given as:

\[
\begin{align*}
\text{High Order rule:} & \quad \bigwedge_{i=1}^{n} (I_{a_i}(x) \land R_{a_i}(y)) \Rightarrow \bigwedge_{j=1}^{m} (I_{d_j}(x) \land R_{d_j}(y)), \\
\end{align*}
\]

where \((x, y) \in U \times U, a_i \in At, \) and \( d_j \in At. \)

### 3.3.3 Quantitative Measures of Rules

The meanings and interpretations of a rule \( \phi \Rightarrow \psi \) can be further clarified by using the extensions \( m(\phi) \) and \( m(\psi) \) of the two concepts. More specifically, we can define quantitative measures indicating the strength of a rule. A systematic analysis of probabilistic quantitative measures can be found in [55]. Two examples of probabilistic quantitative measures are [32]:

\[
\begin{align*}
\text{accuracy}(\phi \Rightarrow \psi) &= \frac{|m(\phi \land \psi)|}{|m(\phi)|}, \text{ and} \\
\text{coverage}(\phi \Rightarrow \psi) &= \frac{|m(\phi \land \psi)|}{|m(\psi)|},
\end{align*}
\]
where $| \cdot |$ denotes the cardinality of a set. The two measures are applicable to both low and high order rules. This demonstrates the flexibility and power of the language $\mathcal{L}$.

While the accuracy reflects the correctness of the rule, the coverage reflects the applicability of the rule. If $\text{accuracy}(\phi \Rightarrow \psi) = 1$, we have a strong association between $\phi$ and $\psi$. A smaller value of accuracy indicates a weaker association. A higher coverage suggests that the relationships of more objects can be derived from the rule. The accuracy and coverage are not independent of each other, one may observe a trade-off between accuracy and coverage. A rule with higher coverage may have a lower accuracy, while a rule with higher accuracy may have a lower coverage.

### 3.4 Rough Set Approaches for the Interpretation of Rules

In this section, the usefulness and flexibility of the language $\mathcal{L}$ is demonstrated by considering two rough set approaches, namely, standard rough set analysis [17, 19, 20] and dominance-based rough set analysis [8–10, 24, 41]. The main differences between the two approaches are their different interpretations of the binary relation $R_a$ of atomic formulas. The basic results are summarized in Table 3.1. For comparison, we also include the results of generalized rough set analysis based on an arbitrary binary relation $R_a$ on attribute values. Note that we use $R$, $E$, and $\succeq$ as qualitative relational symbols to express relationships be-
Table 3.1: Rough Set Approaches for Studying Low and High Order Rules

<table>
<thead>
<tr>
<th>Relation</th>
<th>Low Order Rule</th>
<th>High Order Rule</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>$I_a(x) R_a v_a$</td>
<td>$I_a(x) R_a I_a(y)$</td>
<td>Generalized Rough Set Analysis</td>
</tr>
<tr>
<td></td>
<td>$\Rightarrow I_d(x) R_d v_d$</td>
<td>$\Rightarrow I_d(x) R_d I_d(y)$</td>
<td>Rough Set Analysis</td>
</tr>
<tr>
<td>$E$</td>
<td>$I_a(x) = v_a$</td>
<td>$I_a(x) = I_a(y)$</td>
<td>Standard Rough Set Analysis</td>
</tr>
<tr>
<td></td>
<td>$\Rightarrow I_d(x) = v_d$</td>
<td>$\Rightarrow I_d(x) = I_d(y)$</td>
<td>Rough Set Analysis</td>
</tr>
<tr>
<td>$\succeq$</td>
<td>$I_a(x) \succeq_a v_a$</td>
<td>$I_a(x) \succeq_a I_a(y)$</td>
<td>Dominance-based Rough Set Analysis</td>
</tr>
<tr>
<td></td>
<td>$\Rightarrow I_d(x) \succeq_d v_d$</td>
<td>$\Rightarrow I_d(x) \succeq_d I_d(y)$</td>
<td>Rough Set Analysis</td>
</tr>
</tbody>
</table>

between objects, and $R_a$, $=$, and $\succeq_a$ as quantitative relational symbols to express relationships between their attribute values.

3.4.1 Standard Rough Set Analysis

The indiscernibility relation used in the standard rough set analysis is an equivalence relation on the set of objects [17, 19, 20]. It is used for the extraction of rules for classification and attribute dependency analysis.

Let $E \subseteq U \times U$ be an equivalence relation on $U$, where $U \times U = U^2$ is the Cartesian product of $U$. For a subset of attributes $A \subseteq At$, we can formally define an equivalence relation $U$ as follows:

$$x E y \iff \bigwedge_{a \in A} I_a(x) = I_a(y).$$

That is, $x$ and $y$ are equivalent if and only if $x$ is equivalent to $y$ according to all attributes in $A$. 27
The equivalence relation $E$ partitions $U$ into disjoint subsets known as equivalence classes. That is, the equivalence relation $E$ induces a partitioning of $U$, denoted by $U/E$. Equivalence classes are the basic sets or granules in the universe. We can construct them using atomic formulas of the language $\mathcal{L}$ by replacing the relational symbol from a family of binary relation symbols $\{R_a\}$ to the equality relation $=.

For the first type of granules, we consider objects in the universe $U$ as objects of the domain $D$. The language is given by $\mathcal{L}(\{(a, =, v)\}, U, \{-, \land, \lor, \rightarrow, \leftrightarrow\})$ with atomic formulas of the form of $(a, =, v)$. An object $x \in U$ satisfies an atomic formula $(a, =, v)$ if the value of $x$ on attribute $a$ is $v$, that is, $I_a(x) = v$. We write:

$$x \models (a, =, v) \iff I_a(x) = v.$$ 

The granule corresponding to the atomic formula $(a, =, v)$ is:

$$m(a, =, v) = \{x \in U \mid I_a(x) = v\}.$$ 

The granule $m(a, =, v)$ is also referred to as the block defined by the attribute-value pair $(a, v)$ [11]. The granules corresponding to the compound formulas of the language $\mathcal{L}(\{(a, =, v)\}, U, \{-, \land, \lor, \rightarrow, \leftrightarrow\})$ can be defined by Equation (2.1).

Equivalence classes construct a subsystem of $2^U$ by taking set intersections, unions and complements of equivalence classes. This subsystem is in fact an $\sigma$-algebra of subsets of the universe, written as $\sigma(U/E)$. It contains the empty set $\emptyset$, the entire set $U$, equivalence classes as basic sets, and is closed under set intersection, union and complement. The partition $U/E$ is a basis of $\sigma(U/E)$. The family of
granules is exactly the $\sigma$-algebra $\sigma(U/E)$ [50]. That is:

$$\text{Def}(\mathcal{L}((\{a, =, v\}, U, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\})) = \sigma(U/E).$$

For a set or a granule that is undefinable in the universe $U$, one can approximate it by the lower and upper approximations. For a subset of objects $X \subseteq U$, we define a pair of lower and upper approximation as [50]:

$$\text{apr}(X) = \bigcup \{Y | Y \in \text{Def}(\mathcal{L}((\{a, =, v\}, U, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\})), Y \subseteq X\},$$

$$\overline{\text{apr}}(X) = \bigcap \{Y | Y \in \text{Def}(\mathcal{L}((\{a, =, v\}, U, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\})), X \subseteq Y\}.$$

That is, $\text{apr}(X)$ is the largest definable set contained in $X$, and $\overline{\text{apr}}(X)$ is the smallest definable set containing $X$.

Blocks correspond to the atomic formulas and are used to construct rules. Low order rules can be expressed based on the equality relation $=$. An example of a low order rule is:

$$\bigwedge_{i=1}^{n} (I_{a_{i}}(x) = v_{a_{i}}) \Rightarrow \bigwedge_{j=1}^{m} (I_{d_{j}}(x) = v_{d_{j}}),$$

where $x \in U$, $v_{a_{i}} \in V_{a_{i}}$, $v_{d_{j}} \in V_{d_{j}}$, $a_{i} \in At$, and $d_{j} \in At$.

For the second type of granules, we consider object pairs in two universes $U \times U$ as objects of the domain $D$. Accordingly, one needs to define an equivalence relation on two universes [54]. Let $E_H \subseteq (U \times U)^2$ be an equivalence relation on $U \times U$, where $(U \times U)^2$ is the Cartesian product of $U \times U$. Suppose $E_1$ and $E_2$ are two equivalence relations on $U$. We can formally define an equivalence relation on $U \times U$ as follows:

$$(x_1, y_1)E_H(x_2, y_2) \iff x_1E_1x_2, y_1E_2y_2.$$
That is, object pairs \((x_1, y_1)\) and \((x_2, y_2)\) are equivalent if and only if \(x_1\) is equivalent to \(x_2\) and \(y_1\) is equivalent to \(y_2\).

The equivalence relation \(E_H\) partitions \(U \times U\) into equivalence classes consisting of object pairs. Accordingly, the partition of \(U \times U\) is denoted by \((U \times U)/E_H\). The language of the second type of granule is given by \(\mathcal{L}((\{(a, =)\}, U \times U, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\})\) with atomic formulas of the form of \((a, =)\). A pair of objects \((x, y) \in U \times U\) satisfies an atomic formula \((a, =)\) if the value of \(x\) equals to the value of \(y\) on attribute \(a\), that is, \(I_a(x) = I_a(y)\). We write:

\[(x, y) \models (a, =) \text{ iff } I_a(x) = I_a(y).\]

We rewrite the language as \(\mathcal{L}((\{(a, =)\}, U \times U, \neg, \land, \lor, \rightarrow, \leftrightarrow))\). The granule corresponding to the atomic formula \((a, =)\) is:

\[m(a, =) = \{(x, y) \in U \times U \mid I_a(x) = I_a(y)\}.
\]

The granules corresponding to the compound formulas of the language \(\mathcal{L}((\{(a, =)\}, U \times U, \neg, \land, \lor, \rightarrow, \leftrightarrow))\) can be defined by Equation (2.1). The family of granules is the \(\sigma\)-algebra \(\sigma((U \times U)/E_H)\). That is:

\[\text{Def}(\mathcal{L}((\{(a, =)\}, U \times U, \neg, \land, \lor, \rightarrow, \leftrightarrow})) = \sigma((U \times U)/E_H).
\]

Similarly, for a set or a granule that is undefinable in two universes \(U \times U\), we can define a pair of lower and upper approximation as:

\[
\text{apr}(X) = \bigcup\{Y \mid Y \in \text{Def}(\mathcal{L}((\{(a, =)\}, U \times U, \neg, \land, \lor, \rightarrow, \leftrightarrow))), Y \subseteq X\},
\]

\[
\text{appr}(X) = \bigcap\{Y \mid Y \in \text{Def}(\mathcal{L}((\{(a, =)\}, U \times U, \neg, \land, \lor, \rightarrow, \leftrightarrow))), X \subseteq Y\}.
\]
Table 3.2: An Information Table

<table>
<thead>
<tr>
<th>Object</th>
<th>Height</th>
<th>Hair</th>
<th>Eyes</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>o₁</td>
<td>short</td>
<td>blond</td>
<td>blue</td>
<td>+</td>
</tr>
<tr>
<td>o₂</td>
<td>short</td>
<td>blond</td>
<td>brown</td>
<td>-</td>
</tr>
<tr>
<td>o₃</td>
<td>tall</td>
<td>red</td>
<td>blue</td>
<td>+</td>
</tr>
<tr>
<td>o₄</td>
<td>tall</td>
<td>dark</td>
<td>blue</td>
<td>-</td>
</tr>
<tr>
<td>o₅</td>
<td>tall</td>
<td>dark</td>
<td>blue</td>
<td>-</td>
</tr>
<tr>
<td>o₆</td>
<td>tall</td>
<td>blond</td>
<td>blue</td>
<td>+</td>
</tr>
<tr>
<td>o₇</td>
<td>tall</td>
<td>dark</td>
<td>brown</td>
<td>-</td>
</tr>
<tr>
<td>o₈</td>
<td>short</td>
<td>blond</td>
<td>brown</td>
<td>-</td>
</tr>
</tbody>
</table>

High order rules can be expressed based on the equality relation = accordingly.

An example of a high order rule is:

\[
\bigwedge_{i=1}^{n} (I_{a_i}(x) = I_{a_i}(y)) \Rightarrow \bigwedge_{j=1}^{m} (I_{d_j}(x) = I_{d_j}(y)),
\]

where \((x, y) \in U \times U\), \(a_i \in A_t\), \(d_j \in A_t\).

**Example 1** Table 3.2 is an information table taken from [22]. Each object is described by four attributes. The column labeled by “Class” denotes an expert’s classification of the objects.

An example of a low order rule in Table 3.2 is:

\[
\text{LR}_1: (I_{\text{Hair}}(x) = \text{blond}) \land (I_{\text{Eyes}}(x) = \text{blue}) \Rightarrow (I_{\text{Class}}(x) = +).
\]

That is, if an object has blond hair and blue eyes, then it belongs to class +.
meaning sets of formulas in the rule are:

\[
m(\text{Hair,} =, \text{blond}) = \{o_1, o_2, o_6, o_8\},
\]

\[
m(\text{Eyes,} =, \text{blue}) = \{o_1, o_3, o_4, o_5, o_6\},
\]

\[
m(\text{Class,} =, +) = \{o_1, o_3, o_6\}.
\]

By using the probabilistic quantitative measures, we have:

\[
\text{accuracy}(LR_1) = \frac{|m((\text{Hair,} =, \text{blond}) \land (\text{Eyes,} =, \text{blue}) \land (\text{Class,} =, +))|}{|m((\text{Hair,} =, \text{blond}) \land (\text{Eyes,} =, \text{blue}))|} = 1,
\]

\[
\text{coverage}(LR_1) = \frac{|m((\text{Hair,} =, \text{blond}) \land (\text{Eyes,} =, \text{blue}) \land (\text{Class,} =, +))|}{|m(\text{Class,} =, +)|} = \frac{2}{3}.
\]

The association between \((\text{Hair,} =, \text{blond}) \land (\text{Eyes,} =, \text{blue})\) and \((\text{Class,} =, +)\) reaches the maximum value 1, and the applicability of the rule is also high.

An example of a high order rule in Table 3.2 is:

\[
\text{HR}_1: \left( I_{\text{Height}}(x) = I_{\text{Height}}(y) \right) \land \left( I_{\text{Eyes}}(x) = I_{\text{Eyes}}(y) \right) \Rightarrow \left( I_{\text{Class}}(x) = I_{\text{Class}}(y) \right).
\]

That is, if two objects have the same height and the same eye color, then they
belong to the same class. The meaning sets of formulas in the rule are:

\[
m(\text{Height}, =) = \{(01, 01), (01, 02), (01, 08), (02, 01), (02, 02), (02, 08), (03, 03), (03, 04), (03, 05), (03, 07), (04, 04), (04, 05), (04, 06), (04, 07), (05, 03), (05, 04), (05, 05), (05, 06), (05, 07), (06, 01), (06, 02), (06, 03), (06, 04), (06, 05), (06, 06), (06, 07), (07, 01), (07, 02), (07, 03), (07, 04), (07, 05), (07, 06), (07, 07), (08, 01), (08, 02), (08, 03), (08, 04), (08, 05), (08, 06), (08, 07), (08, 08)\}
\]

\[
m(\text{Eyes}, =) = \{(01, 01), (01, 02), (01, 03), (01, 04), (01, 05), (01, 06), (02, 01), (02, 02), (02, 04), (02, 06), (03, 01), (03, 02), (03, 03), (03, 04), (03, 05), (03, 06), (03, 07), (04, 01), (04, 02), (04, 03), (04, 04), (04, 05), (04, 06), (04, 07), (04, 08), (05, 01), (05, 02), (05, 03), (05, 04), (05, 05), (05, 06), (05, 07), (05, 08), (06, 01), (06, 02), (06, 03), (06, 04), (06, 05), (06, 06), (06, 07), (06, 08), (07, 01), (07, 02), (07, 03), (07, 04), (07, 05), (07, 06), (07, 07), (07, 08), (08, 01), (08, 02), (08, 03), (08, 04), (08, 05), (08, 06), (08, 07), (08, 08)\}
\]

\[
m(\text{Class}, =) = \{(01, 01), (01, 03), (01, 04), (01, 05), (01, 06), (02, 02), (02, 04), (02, 06), (03, 01), (03, 03), (03, 04), (03, 05), (03, 06), (03, 07), (04, 01), (04, 02), (04, 04), (04, 06), (04, 07), (04, 08), (05, 02), (05, 04), (05, 05), (05, 06), (05, 07), (05, 08), (06, 01), (06, 03), (06, 04), (06, 06), (06, 07), (06, 08), (07, 01), (07, 03), (07, 04), (07, 06), (07, 07), (07, 08), (08, 01), (08, 03), (08, 04), (08, 06), (08, 07), (08, 08)\}
\]

By using the probabilistic quantitative measures, we have:

\[
\text{accuracy}(HR_1) = \frac{|m((\text{Height}, =) \land (\text{Eyes}, =) \land (\text{Class}, =))|}{|m((\text{Height}, =) \land (\text{Eyes}, =))|} = \frac{7}{11},
\]

\[
\text{coverage}(HR_1) = \frac{|m((\text{Height}, =) \land (\text{Eyes}, =) \land (\text{Class}, =))|}{|m(\text{Class}, =)|} = \frac{7}{17}.
\]

In this case, \((\text{Height}, =) \land (\text{Eyes}, =)\) does not tell us too much information about the overall object classification in terms of both accuracy and coverage.

The definition of arithmetic operators used in the above example can be found in Tarski's book [31].
3.4.2 Dominance-based Rough Set Analysis

The dominance-based rough set analysis proposed by Greco, Matarazzo and Slowinski [8–10] is based on preference relations on attribute values. It is used for the extraction of rules for ranking and attribute dependency analysis [24, 41, 43].

Let $\succeq \subseteq U \times U$ be a preference relation on $U$. For a subset of attributes $A \subseteq At$, we can formally define a preference relation as follows:

$$x \succeq y \quad \text{iff} \quad \bigwedge_{a \in A} I_a(x) \succeq_a I_a(y).$$

That is, $x$ is ranked ahead of $y$ if and only if $x$ is ranked ahead of $y$ according to all attributes in $A$.

The preference relation induces a covering of the universe, denoted by $U/ \succeq$. A covering is a family of subsets of the universe such that their union is the universe. By allowing a nonempty overlapping of two subsets, a covering is a natural generalization of a partition. For an object $x \in U$, its successor induced by the preference relation $\succeq$ is:

$$(x)_{\succeq} = \{y \mid y \in U, x \succeq y\}.$$

Successors are the basic sets or granules in the universe. The family of all successors is a covering $U/ \succeq$ of the universe and is defined as:

$$U/ \succeq = \{(x)_{\succeq} \mid x \in U\}.$$

Note that the successor is only one type from the neighborhood of all objects of $U$. The covering can also be defined by the family of all predecessors [42]. We can construct successors or predecessors in a covering using atomic formulas of the
language $\mathcal{L}$ by replacing the relational symbol from a family of binary relation symbol $R_a$ to the preference relation $\succeq_a$.

For the first type of granule, we consider objects in the universe as objects of the domain $D$. The language is given by $\mathcal{L}((a, \succeq_a, v)}), U, \{\neg, \land, \lor, \to, \leftrightarrow\})$ with atomic formulas of the form of $(a, \succeq_a, v)$. An object $x \in U$ satisfies an atomic formula $(a, \succeq_a, v)$ if the value of $x$ on attribute $a$ is greater than or equal to $v$, that is, $I_a(x) \succeq_a v$. We write:

\[ x \models (a, \succeq_a, v) \text{ iff } I_a(x) \succeq_a v. \]

We rewrite the language as $\mathcal{L}((a, \succeq_a, v)}), U, \{\neg, \land, \lor, \to, \leftrightarrow\})$. The granule corresponding to the atomic formula $(a, \succeq_a, v)$ is:

\[ m(a, \succeq_a, v) = \{x \in U \mid I_a(x) \succeq_a v\}. \]

Accordingly, the granules corresponding to the compound formulas of the language $\mathcal{L}((a, \succeq_a, v)}), U, \{\neg, \land, \lor, \to, \leftrightarrow\})$ can be defined by Equation (2.1). The family of granules construct a subsystem of $2^U$ by taking set intersections, unions or complements of basic granules, and it can be defined as:

\[
\text{Def}\mathcal{L}(\mathcal{(a, \succeq_a, v)}), U, \{\neg, \land, \lor, \to, \leftrightarrow\})
\]

\[ = \{m(\phi) \mid \phi \in \mathcal{L}(\mathcal{(a, \succeq_a, v)}), U, \{\neg, \land, \lor, \to, \leftrightarrow\})\}. \]

The lower and upper approximation can be defined accordingly for the undefinable sets.

Low order rules can be expressed based on the preference relation. An example
of a low order rule is:

\[ \bigwedge_{i=1}^{n} (I_{a_i}(x) \geq a_i \land v_{a_i}) \Rightarrow \bigwedge_{j=1}^{m} (I_{d_j}(x) \geq d_j \land v_{d_j}) \]

where \( x \in U \), \( v_{a_i} \in V_{a_i}, v_{d_j} \in V_{d_j}, a_i \in At \), and \( d_j \in At \).

For the second type of granule, we need to define a preference relation on two universes. Let \( \succeq_H \subseteq (U \times U)^2 \) be a preference relation on \( U \times U \). Suppose \( \succeq_1 \) and \( \succeq_2 \) are two preference relations on \( U \). We can formally define a preference relation on \( U \times U \) as follows:

\[ (x_1, y_1) \succeq_H (x_2, y_2) \iff x_1 \succeq_1 x_2 \land y_1 \succeq_2 y_2. \]

That is, object pair \((x_1, y_1)\) is ranked ahead of \((x_2, y_2)\) if and only if \(x_1\) is ranked ahead of \(x_2\) and \(y_1\) is ranked ahead of \(y_2\).

The preference relation \( \succeq_H \) partitions \( U \times U \) into subsets consisting of object pairs. Accordingly, the covering of \( U \times U \) is denoted by \((U \times U)/ \succeq_H \). The language of the second type of granule is given by \( \mathcal{L}(\{(a, \geq a)\}, U \times U, \{-, \land, \lor, \rightarrow, \leftarrow\}) \) with atomic formulas of the form of \((a, \geq a)\). A pair of objects \((x, y) \in U \times U\) satisfies an atomic formula \((a, \geq a)\) if the value of \(x\) is greater than or equals to the value of \(y\) on attribute \(a\), that is, \( I_a(x) \geq_a I_a(y) \). We write:

\[ (x, y) \models (a, \geq_a) \text{ iff } I_a(x) \geq_a I_a(y). \]

We rewrite the language as \( \mathcal{L}(\{(a, \geq a)\}, U \times U, \{-, \land, \lor, \rightarrow, \leftarrow\}) \). The granule corresponding to the atomic formula \((a, \geq a)\) is:

\[ m(a, \geq a) = \{(x, y) \in U \times U \mid I_a(x) \geq_a I_a(y)\}. \]
## Table 3.3: An Information Table with Preference Relations

<table>
<thead>
<tr>
<th>Objects</th>
<th>Size</th>
<th>Warranty</th>
<th>Price</th>
<th>Weight</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>middle</td>
<td>3 years</td>
<td>$200</td>
<td>heavy</td>
<td>1</td>
</tr>
<tr>
<td>$p_2$</td>
<td>large</td>
<td>3 years</td>
<td>$300</td>
<td>very heavy</td>
<td>3</td>
</tr>
<tr>
<td>$p_3$</td>
<td>small</td>
<td>3 years</td>
<td>$300</td>
<td>light</td>
<td>3</td>
</tr>
<tr>
<td>$p_4$</td>
<td>small</td>
<td>3 years</td>
<td>$250</td>
<td>very light</td>
<td>2</td>
</tr>
<tr>
<td>$p_5$</td>
<td>small</td>
<td>2 years</td>
<td>$200</td>
<td>very light</td>
<td>3</td>
</tr>
</tbody>
</table>

The granules corresponding to the compound formulas of the language $\mathcal{L} \{ \{ (a, \geq_a) \}, U \times U, \{ \neg, \wedge, \vee, \rightarrow, \leftrightarrow \} \}$ can be defined by Equation (2.1). The family of granules is:

$$
\text{Def}(\mathcal{L} \{ \{ (a, \geq_a) \}, U \times U, \{ \neg, \wedge, \vee, \rightarrow, \leftrightarrow \} \}) = \{ m(\phi) \mid \phi \in \mathcal{L} \{ \{ (a, \geq_a) \}, U \times U, \{ \neg, \wedge, \vee, \rightarrow, \leftrightarrow \} \} \}.
$$

High order rules can be expressed based on the preference relation. An example of a high order rule is:

$$
\bigwedge_{i=1}^{n} (I_{a_i}(x) \geq_a I_{a_i}(y)) \Rightarrow \bigwedge_{j=1}^{m} (I_{d_j}(x) \geq_{d_j} I_{d_j}(y)).
$$

where $(x, y) \in U \times U$, $a_i \in At$, $d_j \in At$.

**Example 2** Table 3.3, taken from [24], is an information table with preference relations on attribute values. It is a group of five products by five manufactures, each product is described by four attributes. The final ranking labeled by “Overall” may be determined by their market share of the products. The preference relations
induce the following orderings:

\(\succ_{\text{Size}}: \) small \(\succ_{\text{Size}}\) middle \(\succ_{\text{Size}}\) large,

\(\succ_{\text{Warranty}}: \) 3 years \(\succ_{\text{Warranty}}\) 2 years,

\(\succ_{\text{Price}}: \) $200 \(\succ_{\text{Price}}\) $250 \(\succ_{\text{Price}}\) $300,

\(\succ_{\text{Weight}}: \) very light \(\succ_{\text{Weight}}\) light \(\succ_{\text{Weight}}\) heavy \(\succ_{\text{Weight}}\) very heavy,

\(\succ_{\text{Overall}}: \) 1 \(\succ_{\text{Overall}}\) 2 \(\succ_{\text{Overall}}\) 3.

An example of a low order rule in this information table is:

\[ LR_2 : (I_{\text{Size}}(x) \succeq_{\text{Size}} \text{middle}) \land (I_{\text{Warranty}}(x) \succeq_{\text{Warranty}} 3 \text{ years}) \Rightarrow I_{\text{Overall}}(x) \succeq_{\text{Overall}} 2. \]

That is, if a product's size is greater than or equal to middle and warranty is greater than or equal to 3 years, then its overall ranking will be greater than or equal to 2. The meaning sets of formulas in the rule are:

\[
m(\text{Size}, \succeq_{\text{Size}}, \text{middle}) = \{p_1, p_3, p_4, p_5\},
\]

\[
m(\text{Warranty}, \succeq_{\text{Warranty}}, 3 \text{ years}) = \{p_1, p_2, p_3, p_4\},
\]

\[
m(\text{Overall}, \succeq_{\text{Overall}}, 2) = \{p_1, p_4\}.
\]

By using the quantitative measures, we have:

\[
\text{accuracy}(LR_2) = \frac{|m(\text{Size}, \succeq_{\text{Size}}, \text{middle}) \land (\text{Warranty}, \succeq_{\text{Warranty}}, 3 \text{ years}) \land (\text{Overall}, \succeq_{\text{Overall}}, 2))|}{|m(\text{Size}, \succeq_{\text{Size}}, \text{middle}) \land (\text{Warranty}, \succeq_{\text{Warranty}}, 3 \text{ years}))|} = \frac{|\{p_1, p_4\}|}{|\{p_1, p_3, p_4\}|} = \frac{2}{3},
\]

\[
\text{coverage}(LR_2) = \frac{|m(\text{Size}, \succeq_{\text{Size}}, \text{middle}) \land (\text{Warranty}, \succeq_{\text{Warranty}}, 3 \text{ years}) \land (\text{Overall}, \succeq_{\text{Overall}}, 2))|}{|m((\text{Overall}, \succeq_{\text{Overall}}, 2))|} = \frac{|\{p_1, p_4\}|}{|\{p_1, p_4\}|} = 1.
\]

There exists a strong association between the two concepts, and applicability of the rule reaches the highest level.
An example of a high order rule is:

$$HR_2 : (I_{\text{Size}}(x) \geq I_{\text{Size}}(y)) \land (I_{\text{Price}}(x) \geq I_{\text{Price}}(y)) \Rightarrow I_{\text{Overall}}(x) \geq I_{\text{Overall}}(y).$$

That is, if one product's size is smaller than or the same as another product and the price is not higher, then this product's overall ranking will be greater than or equal to the other product. The meaning sets of formulas in the rule are:

$$m(\text{Size,} \geq_{\text{Size}}) = \{ (p_1, p_1), (p_1, p_2), (p_2, p_2), (p_3, p_1), (p_3, p_2), (p_3, p_3),$$

$$\ldots$$

$$m(\text{Price,} \geq_{\text{Price}}) = \{ (p_1, p_1), (p_1, p_2), (p_2, p_2), (p_3, p_3), (p_4, p_4), (p_5, p_5),$$

$$\ldots$$

$$m(\text{Overall,} \geq_{\text{Overall}}) = \{ (p_1, p_1), (p_1, p_3), (p_3, p_3), (p_4, p_4), (p_5, p_5), (p_5, p_3),$$

$$\ldots$$

By using the quantitative measures, we have:

$$\text{accuracy}(HR_2) = \frac{|m((\text{Size,} \geq_{\text{Size}}) \land (\text{Price,} \geq_{\text{Price}}) \land (\text{Overall,} \geq_{\text{Overall}}))|}{|m((\text{Size,} \geq_{\text{Size}}) \land (\text{Price,} \geq_{\text{Price}}))|} = 11/13,$$

$$\text{coverage}(HR_2) = \frac{|m((\text{Size,} \geq_{\text{Size}}) \land (\text{Price,} \geq_{\text{Price}}) \land (\text{Overall,} \geq_{\text{Overall}}))|}{|m((\text{Overall,} \geq_{\text{Overall}}))|} = 11/18.$$
3.5 Granular Structures in Rough Set Analysis

We can construct \( \cap \)-closure and \( \cup \)-closure granular structures from an information table. For standard rough set analysis, granular structures can be built using granules in \( \sigma(U/E) \) [56, 59]. Equivalence classes are the building blocks of the granular structures. Formally, we can rewrite the \( \cap \)-closure granular structure by using the equality relation:

\[
G S_\cap(\mathcal{L}) = (Def(\mathcal{L}_\Lambda((a, =, v), U, \{\land\})), \cap),
\]

where \( \mathcal{L}_\Lambda((a, =, v), U, \{\land\}) \) is the sub-language for standard rough set theory with atomic formulas in the form of \((a, =, v)\). Similarly, we can rewrite the \( \cup \)-closure granular structure as:

\[
G S_\cup(\mathcal{L}) = (Def(\mathcal{L}_\lor((a, =, v), U, \{\lor\})), \cup).
\]
Example 3 In Table 3.2, the possible values for three attributes \{Height, Hair, Eyes\} are:

\[
\begin{align*}
V_{\text{Height}} &= \{\text{short, tall}\}, \\
V_{\text{Hair}} &= \{\text{blond, dark, red}\}, \\
V_{\text{Eyes}} &= \{\text{blue, brown}\}.
\end{align*}
\]

If the attribute "Height" is chosen, we can partition the universe into the following equivalence classes or basic granules:

\[
\{o_1, o_2, o_8\}, \{o_3, o_4, o_5, o_6, o_7\},
\]

corresponding to atomic formulas (Height, =, short) and (Height, =, tall), respectively. Similarly, the use of attribute "Hair" produces the following equivalence classes or basic granules:

\[
\{o_1, o_2, o_6, o_8\}, \{o_3\}, \{o_4, o_5, o_7\},
\]

41
corresponding to atomic formulas (Hair, =, blond), (Hair, =, red), and (Hair, =, dark), respectively. For the attribute "Eyes", we have:

\[ \{o_1, o_3, o_4, o_5, o_6\}, \{o_2, o_7, o_8\}, \]

\[ \{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8\}\]

corresponding to atomic formulas (Eyes, =, blue) and (Eyes, =, brown), respectively.

Smaller granules are set intersections of basic granules. For example, sets

\[ \{o_1, o_2, o_8\} \cap \{o_1, o_2, o_6, o_8\} = \{o_1, o_2, o_8\}, \]
\[ \{o_3, o_4, o_5, o_6, o_7\} \cap \{o_4, o_5, o_7\} \cap \{o_2, o_7, o_8\} = \{o_7\}, \]

are smaller granules with the corresponding compound formulas (Height, =, short) \& (Hair, =, blond) and (Height, =, tall) \& (Hair, =, dark) \& (Eyes, =, brown), respectively.

Figure 3.1 draws part of the closure granular structure for Table 3.2. In the figure, we assume that an attribute appears at most once in each formula of GSn(CA). An atomic formula is simply represented by the attribute value. For example, the atomic formula (Height, =, short) is simply written as short.

Larger granules are set unions of basic granules. For example, sets

\[ \{o_1, o_2, o_8\} \cup \{o_4, o_5, o_7\} = \{o_1, o_2, o_4, o_5, o_7, o_8\}, \]
\[ \{o_3, o_4, o_5, o_6, o_7\} \cup \{o_3\} \cup \{o_2, o_7, o_8\} = \{o_2, o_3, o_4, o_5, o_6, o_7, o_8\}, \]

are larger granules for the corresponding compound formulas (Height, =, short) \lor (Hair, =, dark) and (Height, =, tall) \lor (Hair, =, red) \lor (Eyes, =, brown), respectively.
Figure 3.2 draws part of the U-closure granular structure for Table 3.2.
Chapter 4

THE LANGUAGE $\mathcal{L}$ IN FORMAL CONCEPT ANALYSIS

Formal concept analysis [7, 36] studies relationships between objects and their attributes in a formal context. In this chapter, we use formal concept analysis as another concrete granular computing model to further demonstrate the usefulness and flexibility of the language $\mathcal{L}$.

4.1 Formal Contexts

A formal context is a triple $(O, A, R)$ consisting of two sets $O$ and $A$ and a binary relation $R \subseteq O \times A$ between $O$ and $A$. The elements of $O$ are called the objects, and the elements of $A$ are called the attributes that the objects might have. If
(x, a) ∈ R, we say that the object x has the attribute a; we also write it as xRa.

In many situations, attributes are not just properties which objects may or may not have. Attributes such as "Size", "Prize", and "Weight" have values. A formal context with many-valued attributes can be defined as a many-valued context \((O, A, V, R)\) consisting of sets \(O\), \(A\), and \(V\) and a ternary relation \(R \subseteq O \times A \times V\). The elements of \(A\) are many-valued attributes and the elements of \(V\) are their possible attribute values. A many-valued formal context satisfies the following condition:

\[(x, a, v_1) \in R \land (x, a, v_2) \in R \implies (v_1 = v_2).\]

In the language of rough set analysis, one can define a partial map from \(O\) to \(V\) with respect to an attribute \(a\) of a many-valued context \((O, A, V, R)\). If \((x, a, v) \in R\), we write \(I_a(x) = v\).

A many-valued context can be represented by an information table. The rows of the table are the objects and the columns are attributes. The entry in row \(x\) and column \(a\) represents the attribute value \(I_a(x)\). If the attribute \(a\) does not have a value for object \(x\), then there is no entry. A many-valued formal context can be translated into a single-valued context through a process called conceptual scaling. This process contains two essential steps.

In the first step, each attribute of a many-valued context is interpreted into a scale. A scale for the attribute \(a\) of a many-valued context can be defined as a single-valued context \(S_a=(O_a, A_a, R_a)\). The objects of a scale are called scale values and attributes are called scale attributes. The binary relation \(R_a \subseteq O_a \times A_a\) can be interpreted differently according to different types of scales \([7]\). For a better
understanding, we explain three elementary scales in a simple example.

**Example 4** Table 5.1 is an example of a many-valued context for some televisions. The scale contexts of three attributes, “Type”, “Clarity”, and “Price” are given in Table 4.2.

The first type of scale is called nominal scale. It is used to scale attributes with the values that mutually exclude each other. The attribute “Type” with the values \{CRT, LCD, Plasma\} in Table 4.2 uses this kind of scale.

The second type of scale is called an ordinal scale. It is used to scale attributes with the values that are ordered and each value implies the weak ones. The attribute “Clarity” with the values \{clear, very clear, extremely clear\} in Table 4.2 uses this kind of scale.

The third type of scale is called interordinal scale. It is used to scale attributes with the values that have bipolar orderings. The attribute “Price” with the values \{$0 \leq Price < $1000, $1000 \leq Price < $3000, $3000 \leq Price < $4000$\} in Table 4.2 uses this kind of scale.

The second step of conceptual scaling is to join the scales together to make
Table 4.2: Examples of Scale Contexts

<table>
<thead>
<tr>
<th>S_{Type}</th>
<th>CRT</th>
<th>CRT</th>
<th>Plasma</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRT</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCD</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Plasma</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S_{Clarity}</th>
<th>≤ clear</th>
<th>≤ very clear</th>
<th>≤ extremely clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>very clear</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>extremely clear</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S_{Price}</th>
<th>cheap</th>
<th>mid-range</th>
<th>expensive</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 \leq \text{Price} &lt; $1000</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$1000 \leq \text{Price} &lt; $3000</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>$3000 \leq \text{Price} &lt; $4000</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

A single-valued context. In the case of plain scaling, the object set $O$ of the many-valued context remains unchanged, every many-valued attribute $a$ is replaced by the scale attributes of the scale $(O_a, A_a, R_a)$. The attribute set of the derived single-valued context is the disjoint union of scale attribute sets. We can rewrite the scale attribute set $A_a$ to $\hat{A}_a=\{a\}\times A_a$ to ensure that those scale attribute sets are disjoint. Therefore, the derived single-valued formal context from many-valued context $(O, A, V, R)$ with respect to plain scaling can be defined as
Table 4.3: Derived Single-valued Context

a: CRT; b: LCD; c: Plasma; d: ≤ clear; e: ≤ very clear; 
f: ≤ extremely clear; g: cheap; h: mid-range; i: expensive.

<table>
<thead>
<tr>
<th></th>
<th>Type</th>
<th>Clarity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>tvi</td>
<td>v</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tv1</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>tv2</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>tv3</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>tv4</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

That is, an object \( x \in O \) has the attribute value \( b \in B \) with respect to the attribute \( a \in A \) in the context \((O, B, R')\) if and only if \( x \) has the attribute value \( v \) with respect to attribute \( a \) in the many-valued context \((O, A, V, R)\), and \( v \) is \( R_a \)-related to \( b \) in the scale context \( S_a \). Table 4.3 shows the derived single-valued context by combining the many-valued context of Table 5.1 and the scale contexts of Table 4.2.
4.2 Granules in Formal Concept Analysis

The processes of defining granules and granular structures are different in rough set analysis and formal concept analysis. They build granules and granular structures based on two different interpretations of the notion of definability. In rough set analysis, a definable granule is the union of some equivalence classes. In formal concept analysis, one is interested in granules of objects that are extensions of formal concepts. A concept derived from a formal context is a pair of a set of objects and a set of attributes, called the extension and intension of the concept. Furthermore, the extension and intension are mutually definable, that is, the intension of the extension is the extension of the intension and vice versa.

For a many-valued formal context \((O, A, V, R)\), one first needs to transfer this many-valued context into a single-valued context \((O, B, R')\). The formal concepts constructed from this derived single-valued context are then interpreted as the formal concepts of the many-valued context.

Since a many-valued formal context \((O, A, V, R)\) can be translated into a single-valued context \((O, B, R')\), it is sufficient to consider the construction process of formal concepts in \((O, B, R')\). We can construct the language \(L\) by using a single-valued formal context as the model \(M\), \(O\) as the domain \(D\), and the set of atomic formulas is given by \(A = \{b \mid b \in B\}\). That is, the atomic formula is denoted as the attribute value \((b)\) or simply \(b\). An object \(x \in O\) satisfies an atomic formula \(b\) if the object has the attribute \(b\):

\[ x \models b \iff xR'\).

49
We rewrite the language as $\mathcal{L}(\{b\}, O, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\})$. In Table 4.3, examples of atomic formulas are (Type, CRT) and (Clarity, ≤very clear). For simplicity, we also write them as “CRT” and “≤very clear”, respectively.

By using the language $\mathcal{L}$, we can formally define a formal concept in a formal context. For a set $X \subseteq O$ of objects, we define a mapping $O \rightarrow A$ as:

$$m'(X) = \{b \in A \mid x \models b \text{ for all } x \in X\},$$

that is, the set of atomic formulas satisfied by all the objects in $X$. Correspondingly, for a set $P \subseteq A$ of atomic formulas, we define a mapping $A \rightarrow O$ as:

$$m(P) = \{x \in O \mid x \models \bigwedge_{b \in P} b\},$$

that is, the set of objects satisfies all the atomic formulas in $P$. With these two mappings defined, a formal concept can be defined as a pair satisfying the condition:

$$(X, P) = \{X \subseteq O, P \subseteq A \mid X = m(P), P = m'(X)\}.$$ 

The set of objects $X=\text{extent}(X, P)$ is the extension of the formal concept, and the set of atomic formulas $P=\text{intent}(X, P)$ is the intension of the formal concept. By using logic formulas, we can rewrite a formal concept as $(X, \bigwedge_{b \in P} b)$.

For an atomic formula $b \in A$, we can derive a formal concept $(m(\{b\}), m'(m(\{b\})))$. A subset or a granule $X \subseteq O$ is definable in a formal context if and only if there exits a subset of atomic formulas $P \subseteq A$ in the language $\mathcal{L}(\{b\}, O, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\})$ such that $(X, P)$ is a formal concept, that is, $X = 50.$
\( m(P) \) and \( P = m'(X) \). Otherwise, it is undefinable. Let \( \mathcal{B}(O, B, R') \) or simply \( \mathcal{B} \) denote the set of all formal concepts of the context. The family of definable granules is given by:

\[
\text{Def}(\{b\}, O, \{\neg, \land, \lor, \rightarrow, \leftrightarrow\}) = \{\text{extent}(X, P) \mid (X, P) \in \mathcal{B}\}.
\]

If a formal context \( (O, B, R') \) is treated as a binary information table, one can easily observe a close relationship between rough set analysis and formal concept analysis. A definable set of objects in formal concept analysis is a definable set in rough set analysis, but the reverse is not true.

### 4.3 Granular Structures in Formal Concept Analysis

The family of all formal concepts forms a complete lattice called a concept lattice, through which the relationships between formal concepts can be visualized [7]. The meet and join of the lattice are defined based on the set-theoretic operators of intersection (\( \cap \)), union (\( \cup \)) and the mappings between the object set \( O \) and atomic formula set \( A \), written as:

\[
(X_1, P_1) \land (X_2, P_2) = (X_1 \cap X_2, m'(m(P_1 \cup P_2))),
\]

\[
(X_1, P_1) \lor (X_2, P_2) = (m(m'(X_1 \cup X_2)), P_1 \cap P_2).
\]

The order relation of the concept lattice can be defined as follows. For two formal concepts \( (X_1, P_1) \) and \( (X_2, P_2) \), if \( X_1 \subseteq X_2 \) (which is equivalent to \( P_2 \subseteq P_1 \)), then
(X_2, P_2) is a superconcept of (X_1, P_1) and (X_1, P_1) is a subconcept of (X_2, P_2), written as (X_1, P_1) \leq (X_2, P_2).

From the viewpoint of granular computing, the extensions of superconcepts are larger granules which may be decomposed into smaller granules as extensions of subconcepts. We consider a concept lattice as an \( \cap \)-closure granular structure which only includes granules that are extensions of formal concepts. Each granule in a concept lattice can be labeled by formulas of the language \( \mathcal{L}_\Lambda \).

**Example 5** The process of forming a concept lattice from Table 4.3 can be illustrated as follows. If attribute \( \text{"clear"} \) is chosen, objects satisfying this attribute construct the most general granule which includes all the objects of this context.

The corresponding formal concept is \( (\{tv1, tv2, tv3, tv4\}, \leq \text{clear}) \), where \( \leq \text{clear} \) is an atomic formula. The second level are the granules whose intensions only
include the conjunction of two atomic formulas. The corresponding formal concept is \( \{tv_2, tv_3, tv_4\}, \leq \text{clear} \land \leq \text{very clear} \). The third level are the granules whose intensions only include the conjunction of three atomic formulas. The corresponding formal concept is \( \{tv_1\}, \text{CRT} \land \leq \text{clear} \land \text{cheap} \), \( \{tv_2, tv_4\}, \text{LCD} \land \leq \text{clear} \land \leq \text{very clear} \) and \( \{tv_3, tv_4\}, \leq \text{clear} \land \leq \text{very clear} \land \text{expensive} \). The intersections of granules produce the smaller granules in the fourth level and so on. For example, the set,
\[
\{tv_2, tv_4\} \cap \{tv_3, tv_4\} = \{tv_4\},
\]
is a smaller granule corresponding to the formal concept \( \{tv_4\}, \text{LCD} \land \leq \text{clear} \land \leq \text{very clear} \land \text{expensive} \). The most specific granule is the empty set \( \emptyset \) corresponding to the conjunction of all atomic formulas as its intension.

The line diagram in Figure 4.1 represents the concept lattice of Table 4.3 and includes 9 formal concepts. The intensions of each formal concept are labeled by formulas of the language \( \mathcal{L}_A \).

4.4 Implications and Dependencies in Formal Concept Analysis

Relationships between attributes of a formal context can be automatically generated from a concept lattice. There are two types of these relationships, called implications and dependencies [7]. An implication expresses relationships between attribute values of objects in a formal context, which is a typical example
of low order rules introduced in section 3.3. An example of the implications in Figure 4.1 is, “every object that has attribute expensive also has attribute \( \leq \text{clear}, \) so expensive implies \( \leq \text{clear}. \)” A dependency expresses relationships between object pairs with regards to certain attributes in a formal concept, which is a typical example of high order rules introduced in section 3.3. An example of dependencies in Figure 4.1 is, “if two objects have the same Type, then they must have the same Clarity, so Clarity depends on Type.” In general, dependencies can be seen as an abstraction of implications. Identifying these relationships from large databases is a well-known problem in the field of knowledge discovery. The introduction of formal concept analysis provides an important mathematical methodology for addressing this problem.

We express implications and dependencies by using formulas of the language \( \mathcal{L} \).

4.4.1 Implications

An implication can be written as \( P \Rightarrow C \) for a formal context \( (O, B, R') \), where \( P \) and \( C \) are subsets of attribute set \( B \). The set \( P \) is called the premise and \( C \) is called the conclusion of this implication. The implication \( P \Rightarrow C \) holds in the context \( (O, B, R') \) if objects having attributes in \( P \) also have attributes in \( C \). An example of an implication is given based on the satisfiability of atomic formulas defined in section 4.2:

\[
\bigwedge_{i=1}^{n}(x \ R' \ p_i) \Rightarrow \bigwedge_{j=1}^{m}(x \ R' \ c_j),
\]
where $p_i \in P$ and $c_j \in C$. Note that an arbitrary attribute set $P \subseteq B$ or $C \subseteq B$ is not necessarily the intensions of some formal concepts.

### 4.4.2 Functional and Ordinal Dependencies

An dependency between attributes can be written as $H \Rightarrow K$ for a many-valued context $(O, A, V, R)$, where $H$ and $K$ are subsets of an attribute set $A$. The dependency $H \Rightarrow K$ holds in the context $(O, A, V, R)$ if two objects have the same attribute values with respect to all attributes in $H$. The same must also be true for the attributes in $K$. An example of a dependency is given by following the way of interpreting high order rules of an information table:

$$
\bigwedge_{i=1}^{n} (I_{a_i}(x) \, R_{a_i} \, I_{a_i}(y)) \Rightarrow \bigwedge_{j=1}^{m} (I_{d_j}(x) \, R_{d_j} \, I_{d_j}(y)),
$$

where $(x, y) \in O \times O$, $a_i \in H$ and $d_i \in K$.

There are different types of dependencies between attributes with regards to the different interpretations of the binary relation $R_a$ [7]. If $R_a$ is interpreted as the equality relation $=$, then $K$ is functionally dependent on $H$ if the following dependency holds for every pair of objects $(x, y) \in O \times O$:

$$
\bigwedge_{i=1}^{n} (I_{a_i}(x) = I_{a_i}(y)) \Rightarrow \bigwedge_{j=1}^{m} (I_{d_j}(x) = I_{d_j}(y)).
$$

If $R_a$ is interpreted as an order relation $\geq_a$, then $K$ is ordinally dependent on $H$ if the following dependency holds for every pair of objects $(x, y) \in O \times O$:

$$
\bigwedge_{i=1}^{n} (I_{a_i}(x) \geq_a I_{a_i}(y)) \Rightarrow \bigwedge_{j=1}^{m} (I_{d_j}(x) \geq_d I_{d_j}(y)).
$$
4.4.3 Quantitative Measures of Implications and Dependencies

Implications and dependencies are simple forms of regularities that can be used for the reasoning about knowledge. In a practical case, there may be too many irrelevant implications or dependencies existing in a context since they will be held even if there is only one object that satisfies the rule. To alleviate this problem, quantitative measures can be used to evaluate the statistical significance of the rules that hold. One may choose a threshold value and only select those rules whose measures are above the threshold value. The quantitative measures of mining association rules can be used to measure the strength of implications and dependencies [1].

There are two types of quantitative measures based on a threshold value, called the support and confidence measures. By using the language $\mathcal{L}$, the meanings and interpretations of an implication or dependency $\phi \Rightarrow \psi$ can be further clarified by using the extensions $m(\phi)$ and $m(\psi)$ of the two concepts. We can formally interpret these two measures in formal concept analysis as:

$$support(\phi \Rightarrow \psi) = \frac{|m(\phi \land \psi)|}{|O|},$$

$$confidence(\phi \Rightarrow \psi) = \frac{|m(\phi \land \psi)|}{|m(\phi)|},$$

where $|\cdot|$ denotes the cardinality of a set and $|O|$ denotes the number of objects in a context. The two measures are applicable to both implications and dependencies. The implication or dependency holds in a context if the value of
support or confidence is above the threshold $\text{minsupp} \in [0, 1]$ or $\text{minconf} \in [0, 1]$.

This demonstrates the flexibility and power of the language $L$.

**Example 6** An example of an implication in Table 4.3 is:

\[ IR : (x \ R' \ \text{LCD}) \land (x \ R' \ \leq \text{veryclear}) \Rightarrow (x \ R' \ \text{expensive}). \]

That is, if a tv has LCD as its Type, and it's Clarity is very clear or less, then it's Price is expensive. The meaning sets of formulas in the implication are:

- $m(\text{LCD}) = \{tv_2, tv_4\}$,
- $m(\leq \text{veryclear}) = \{tv_2, tv_3, tv_4\}$,
- $m(\text{expensive}) = \{tv_3, tv_4\}$.

By using the quantitative measures, we have:

\[
\begin{align*}
\text{support}(IR) &= \frac{|m(\text{LCD} \land \leq \text{veryclear} \land \text{expensive})|}{|O|} \\
&= \frac{|\{tv_4\}|}{|O|} = 1/4,
\end{align*}
\]

\[
\begin{align*}
\text{coverage}(IR) &= \frac{|m(\text{LCD} \land \leq \text{veryclear} \land \text{expensive})|}{|m(\text{LCD} \land \leq \text{veryclear})|} \\
&= \frac{|\{tv_4\}|}{|\{tv_2, tv_4\}|} = 1/2.
\end{align*}
\]

If we assume that the $\text{minsupp} = \text{minconf} = 80\%$, then neither support nor the confidence of this implication is above the threshold value.

An example of a functional dependency is:

\[ DR : (I_{\text{Type}}(x) = I_{\text{Type}}(y)) \land (I_{\text{Clarity}}(x) = I_{\text{Clarity}}(y)) \Rightarrow (I_{\text{Price}}(x) = I_{\text{Price}}(y)). \]
That is, if two objects have the same Type and Clarity, then their Price range will be the same. The meaning sets of formulas in the implication are:

\[
m(\text{Type}, =) = \{(tv_1, tv_1), (tv_2, tv_2), (tv_3, tv_3), (tv_4, tv_4), (tv_4, tv_2)\},
\]
\[
m(\text{Clarity}, =) = \{(tv_1, tv_1), (tv_2, tv_2), (tv_2, tv_4), (tv_3, tv_3), (tv_4, tv_4), (tv_4, tv_2)\},
\]
\[
m(\text{Price}, =) = \{(tv_1, tv_1), (tv_2, tv_2), (tv_3, tv_3), (tv_3, tv_4), (tv_4, tv_4), (tv_4, tv_3)\}.
\]

By using the quantitative measures, we have:

\[
support(DR) = \frac{|m((\text{Type}, =) \land (\text{Clarity}, =) \land (\text{Price}, =))|}{|O \times O|} = 1/4,
\]
\[
confidence(DR) = \frac{|m((\text{Type}, =) \land (\text{Clarity}, =))|}{|m((\text{Type}, =) \land (\text{Clarity}, =))|} = 2/3.
\]

In this case, if we assume that the minsup = minconf = 60%, then only the confidence of this functional dependency is above the threshold value.
Chapter 5

CONCLUSION AND FUTURE RESEARCH

This thesis presented a logic-based study of granular computing. We focused on the theoretical foundations that enable us to explain some fundamental issues of granular computing.

5.1 Summary

Granular computing models human problem solving with different-sized grains of knowledge. Fundamental notions of granular computing are basic granules, granular structures and rules. Basic granules are the elementary units of granular computing and represent the basic pieces of knowledge. A granular structure provides a structured description of a system or an application under discussion. Rules, as a commonly used form for representing knowledge, can be thought of as
relationships between granules. A fundamental issue of granular computing is the representation, construction and utilization of these basic notions. In this thesis, we address this issue by proposing a logic approach to precisely describe the definability of basic granules and to construct the granular structures and rules. The tasks of granular computing are formalized during this process. Different granular computing models can be analyzed within this logic framework.

We introduce a logic language $\mathcal{L}$. Atomic formulas correspond to basic granules. Other formulas of the language are recursively constructed from a set of atomic formulas. The meaning of a formula is defined in Tarski's style semantics by using the model $\mathcal{M} = (D, K)$. Based on the knowledge $K$ of the model, it is assumed that an object in the domain $D$ either satisfies a formula or does not satisfy a formula. A granule is jointly described by a pair $(m(\phi), \phi)$ consisting of a subset $m(\phi)$ of the domain $D$ and a formula $\phi$ of the language $\mathcal{L}$. The main difference between $\mathcal{L}$ and other existing decision logic languages is the treatment of atomic formulas. Instead of defining an atomic formula specifically as an (attribute, value) pair [19] or an (attribute, relation, value) triplet [53], we postpone the definition of atomic formulas to particular applications. That is, we build the language based on a set of atomic formulas without giving them a concrete physical meaning. This flexibility enables us to interpret granules and granular structures in different applications. Furthermore, the notions of indiscernibility and definability are formally defined. Two sub-languages of $\mathcal{L}$ are introduced, each of them using only one logic connective. They lead to the introduction of two types of granular structures, namely, $\cap$-closure and $\cup$-closure granular struc-
tures. Different types of rules can be generated from different types of granules, and rule learning algorithms can be designed based on granular structures.

Depending on the particular application, we can interpret the language by using different types of atomic formulas and the associated models. The flexibility of the language $\mathcal{L}$ is demonstrated by considering two concrete granular computing models, namely, rough set analysis and formal concept analysis. These two theories study object-attribute relationships in a data table. They share high-level similarities and can be interpreted uniformly by $\mathcal{L}$. The main differences between them are their choices of definable granules and granular structures. The unified study of these two theories demonstrates the potential of $\mathcal{L}$ for granular computing. Table 5.1 shows the unified interpretation of rough set analysis and formal concept analysis. We suggest that one may study data analysis and rule mining in a much wider class of granular computing models. Algorithms and evaluation measures can be designed uniformly for these models.

### 5.2 Future Research

The work presented in this thesis establishes a logical foundation of granular computing. As an important and comprehensive extension of this thesis, we can develop potentially useful methodologies and algorithms to mine granular information and knowledge structures from data to enhance the applications of granular computing.

A single rule represents the relationship between two granules and a set of
<table>
<thead>
<tr>
<th>Rough Set Analysis</th>
<th>Formal Concept Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic notions</strong></td>
<td><strong>Interpretations</strong></td>
</tr>
<tr>
<td>Information tables</td>
<td>$\mathcal{M} = (U, K)$</td>
</tr>
<tr>
<td>Indiscernibility relation</td>
<td>$x \sim_{\mathcal{A}_0} y$ iff $m'(x) \cap \mathcal{A}_0 = m'(y) \cap \mathcal{A}_0$</td>
</tr>
<tr>
<td>Basic granules</td>
<td>$(a, R_a, v)$</td>
</tr>
<tr>
<td>Definable granules</td>
<td>$X = m(\phi)$</td>
</tr>
<tr>
<td>Granular structures</td>
<td>$GS_n(\mathcal{L})$</td>
</tr>
<tr>
<td>Low-order rules</td>
<td>$\bigwedge_{i=1}^n (I_{a_i}(x) R_{a_i} v_{a_i}) \implies \bigwedge_{i=1}^m (I_{d_j}(x) R_{d_j} v_{d_j})$</td>
</tr>
<tr>
<td>High-order rules</td>
<td>$\bigwedge_{i=1}^n (I_{a_i}(x) R_{a_i} I_{a_i}(y)) \implies \bigwedge_{j=1}^m (I_{d_j}(x) R_{d_j} I_{d_j}(y))$</td>
</tr>
</tbody>
</table>

Inter-related rules represent a granular structure. In order to understand such structures, we need to design measures to characterize rules and rules sets. The strength of a rule is measured by some evaluation criteria. The entire system performance is based on a group of rules. It is necessary to find the criteria for measuring a group of rules and to analyze the relationships between these criteria and their relationships with single rule measures. Moreover, we should
take the uncertainty factor of rule measures into consideration. The results will enable us to properly measure rules and rule sets, which in turn will help us have qualitative and quantitative characterizations of granular information and knowledge structures.

In summary, the future work can aim at a general research strategy that covers the mining of granular information and knowledge structures, and applies the discovered knowledge to real-world applications. It is complimentary to existing studies of data mining that do not pay sufficient attention to multiple levels of granularity.
Bibliography


[16] Murphy, G.L. Theories and concept formation, in: Categories and Concepts,
Theoretical Views and Inductive Data Analysis, Academic Press, New York,

set approach, Computational Intelligence, 17, 514-544, 2001.

[18] Ogden, C.K., Richards I.A. The Meaning of Meaning: a Study of the In-
fluence of Language upon Thought and of the Science of Symbolism, 8th

[19] Pawlak, Z. Rough Sets - Theoretical Aspects of Reasoning about Data,

[20] Pawlak, Z., Skowron, A. Rough sets: some extensions, Information Science,

[21] Peikoff, L. Objectivism: the Philosophy of Ayn Rand, Dutton, New York,

[22] Quinlan, J. R. Learning efficient classification procedures and their appli-
cation to chess end-games, in: Machine Learning: An Artificial Intelligence


71