Document Management and Retrieval for Specialised Domains: An Evolutionary User-Based Approach

Mihye Kim

A thesis submitted to
The School of Computer Science and Engineering
The University of New South Wales
Sydney Australia

in fulfilment of the requirements for the degree of
Doctor of Philosophy

March 2003
Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgment is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project’s design and conception or in style, presentation and linguistic expression is acknowledged.

(Signed) Mihye Kim 17/07/2003
Abstract

Browsing marked-up documents by traversing hyperlinks has become probably the most important means by which documents are accessed, both via the World Wide Web (WWW) and organisational Intranets. However, there is a pressing demand for document management and retrieval systems to deal appropriately with the massive number of documents available. There are two classes of solution: general search engines, whether for the WWW or an Intranet, which make little use of specific domain knowledge or hand-crafted specialised systems which are costly to build and maintain.

The aim of this thesis was to develop a document management and retrieval system suitable for small communities as well as individuals in specialised domains on the Web. The aim was to allow users to easily create and maintain their own organisation of documents while ensuring continual improvement in the retrieval performance of the system as it evolves. The system developed is based on the free annotation of documents by users and is browsed using the concept lattice of Formal Concept Analysis (FCA). A number of annotation support tools were developed to aid the annotation process so that a suitable system evolved. Experiments were conducted in using the system to assist in finding staff and student home pages at the School of Computer Science and Engineering, University of New South Wales.

Results indicated that the annotation tools provided a good level of assistance so that documents were easily organised and a lattice-based browsing structure that evolves in an ad hoc fashion provided good efficiency in retrieval performance. An interesting result suggested that although an established external taxonomy can be useful in proposing annotation terms, users appear to be very selective in their use of terms proposed. Results also supported the hypothesis that the concept lattice of FCA helped take users beyond a narrow search to find other useful documents. In general, lattice-based browsing was considered as a more helpful method than Boolean queries or hierarchical browsing for searching a specialised domain.

We conclude that the concept lattice of Formal Concept Analysis, supported by annotation techniques is a useful way of supporting the flexible open management of documents required by individuals, small communities and in specialised domains. It seems likely that this approach can be readily integrated with other developments such as further improvements in search engines and the use of semantically marked-up documents, and provide a unique advantage in supporting autonomous management of documents by individuals and groups – in a way that is closely aligned with the autonomy of the WWW.
Acknowledgments

There are many people I should thank who helped me to come to this moment. First of all, I would like to express my special gratitude to my supervisor Prof. Paul Compton for his guidance, ideas and insights towards this thesis. I am also grateful for his financial support, encouragement and help. This thesis would not have been possible without him. I must also thank Dr ByeongHo Kang who suggested this project to me. He sowed the seed of this thesis.

I would like to thank staff and research students who participated in the experimental study. Their contribution to this thesis is invaluable.

I would like to especially thank Dr Rex Kwok for his generous time in discussing and reading drafts, and offering suggestions and encouragement. I would also like to thank Dr Bao Vo for helping in formalising of mathematical formulas, and Pamela Mort and Victor Jauregui for their feedback on drafts. In particular, I thank Jane Brennan for sharing and encouraging with lots of love and friendship. Additionally, I would like to thank the following people for their support, help and friendship: A/Prof. Achim Hoffmann, Prof. Norman Foo, Dr Rodrigo Martinez-Bejar, Dr Ashesh J. Mahidadia, Dr YoungJu Yho, Hendra Suryanto, SeungYeol Yoo, Tri Minh Cao, Son Bao Pham, Julian Kerr, Sue Lewis, Angela Finlayson and Abdus Khan. Many thanks are also due to many people in the School of Computer Science and Engineering, the University of New South Wales, Australia.

I am especially grateful to all my friends Veronica, Gemma, Cecilia, Lucy and her husband Stephen, Maria, Ameleta and her husband Albino for their much love and prayer. I would also like to give my special thanks to Fr. Augustine for his love, prayer and encouragement. I must thank my family - my father and mother, brothers and sisters, nieces and nephews - for their endless love, prayer and emotional and financial support.

I would like to express my gratitude to all members of my community AFI for their love and support. In particular, I am grateful for their allowance and acceptance of my study for such a long time living alone here in Australia.

Finally, I thank you - Lord and Holy Mary - for Your endless Love and Grace which allowed me to persevere, and thank you for Your Companionship in moments of uncertainty, anxiety and conflict during all my long journey.
I dedicate this thesis to Loving God.
# Table of Contents

**Chapter 1** Introduction
1.1 Document Management and Retrieval for Specialised Domains ........................................ 1  
1.2 The Aim of this Thesis ........................................................................................................... 4  
1.3 The Structure of this Thesis .................................................................................................. 6  

**Chapter 2** Document Management and Retrieval in Literature ........................................ 8  
2.1 General Approach ................................................................................................................. 9  
2.1.1 Boolean Query ................................................................................................................ 10  
2.1.2 Clustering ....................................................................................................................... 12  
2.2 Ontological Approach .......................................................................................................... 13  
2.2.1 A Notion of Ontology ..................................................................................................... 18  
2.2.2 Types of Ontologies ....................................................................................................... 20  
2.2.3 The Issues relevant to Ontologies .................................................................................. 22  
2.3 Formal Concept Analysis Approach .................................................................................... 27  
2.4 Proposed Approach ............................................................................................................. 29  
2.5 Chapter Summary ................................................................................................................. 30  

**Chapter 3** Document Management for Retrieval with Ripple-Down Rules .......................... 31  
3.1 Ripple-Down Rules .............................................................................................................. 32  
3.1.1 Background of RDR ...................................................................................................... 32  
3.1.2 Basics of RDR ............................................................................................................... 33  
3.1.3 Strengths of RDR .......................................................................................................... 35  
3.1.4 Limitations of RDR ........................................................................................................ 36  
3.2 A Help Desk System with Ripple-Down Rules ................................................................. 37  
3.2.1 Overview of the System ............................................................................................... 38  
3.2.2 Keywords and Help Questions ...................................................................................... 39  
3.2.3 Knowledge Structure .................................................................................................... 40  
3.2.4 Knowledge Acquisition ................................................................................................. 41  
3.2.5 Search Methods ............................................................................................................ 42  
3.2.6 Optimising Process of a Rule Tree ................................................................................. 44  
3.3 Conclusion and Discussion .................................................................................................. 46  
3.4 Chapter Summary ............................................................................................................... 49  

**Chapter 4** Formal Concept Analysis .................................................................................... 51  
4.1 Basis Notions of FCA .......................................................................................................... 52  
4.1.1 Formal Context .............................................................................................................. 52  
4.1.2 Formal Concept .............................................................................................................. 53  
4.2 Concept Lattice .................................................................................................................. 54  
4.2.1 Construction of a Concept Lattice ................................................................................ 54  
4.2.2 Algorithms for Constructing a Concept Lattice ............................................................ 56  
4.3 Conceptual Scaling ............................................................................................................. 58  
4.4 FCA for Information Retrieval ............................................................................................ 62  
4.4.1 Godin et al. Approach ................................................................................................. 63  
4.4.2 Carpineto and Romano Approach ............................................................................... 64
4.4.3 FaIR Approach............................................................................................................. 64
4.4.4 Cole et al. Approach.................................................................................................. 65
4.4.5 Proposed Approach................................................................................................. 66
4.5 Chapter Summary ...................................................................................................... 71

Chapter 5 A Formal Framework of
Document Management and Retrieval for Specialised Domains................................. 72
5.1 Basic Notions of the System ...................................................................................... 73
  5.1.1 Formal Context ..................................................................................................... 73
  5.1.2 Formal Concept .................................................................................................. 74
  5.1.3 Concept Lattice ................................................................................................. 75
5.2 Incremental Construction of a Concept Lattice ......................................................... 76
  5.2.1 Basic Definitions of the Algorithms ................................................................... 76
  5.2.2 Description of the Algorithms .......................................................................... 77
5.3 Document Management ............................................................................................ 82
  5.3.1 Phase One: Reusing Terms in the System ...................................................... 83
  5.3.2 Phase Two: Using Imported Terms from Taxonomies .................................. 83
  5.3.3 Phase Three: Using co-occurred Terms in the Lattice ................................... 87
  5.3.4 Phase Four: Identifying related Documents .................................................... 91
  5.3.5 Phase Five: Adding New Terms ...................................................................... 92
  5.3.6 Phase Six: Logging Users’ Queries .................................................................. 93
5.4 Document Retrieval ................................................................................................ 94
  5.4.1 Browsing the Lattice Structure ........................................................................ 96
  5.4.2 Entering a Boolean Query ................................................................................ 96
5.5 Conceptual Scaling ................................................................................................. 98
  5.5.1 Conceptual Scaling for a Many-valued Context .............................................. 100
  5.5.2 Conceptual Scaling for a One-valued Context ................................................ 103
5.6 Chapter Summary .................................................................................................... 107

Chapter 6 Implementation ............................................................................................. 109
6.1 Overview of the System ........................................................................................ 110
6.2 Basic Environment of the System .......................................................................... 113
6.3 Presentation of the System ..................................................................................... 114
  6.3.1 Domain of Research Interests in a Computer Science School ...................... 114
    6.3.1.1 Document Annotation .............................................................................. 114
    6.3.1.2 System Maintenance by a Knowledge Engineer ...................................... 121
    6.3.1.3 Document Retrieval and Browsing ......................................................... 123
  6.3.2 Domain of Proceedings Papers ....................................................................... 131
6.4 Chapter Summary .................................................................................................... 134

Chapter 7 Experimental Evaluation ............................................................................. 136
7.1 Experimental Design............................................................................................... 137
7.2 Experimental Results ............................................................................................. 138
  7.2.1 Annotation Mechanisms .................................................................................. 138
    7.2.1.1 Users’ Annotation Activities .................................................................. 138
    7.2.1.2 Survey: Questionnaire on the Annotation Mechanisms ...................... 143
  7.2.2 Ontology Evolution ......................................................................................... 149
  7.2.3 Lattice-based Browsing .................................................................................... 155
List of Figures

Figure 2.1. An infrastructure of the Semantic Web ................................................................. 14
Figure 2.2. An instantiation example of the ontologies (an annotated home page) ............... 16
Figure 2.3. A search result using the ontological browser of KA² ........................................ 17
Figure 2.4. Top-level categories of Cyc (adapted from Lenat and Guha 1990) ....................... 21

Figure 3.1. An example of the knowledge structure for the help system .................................. 40
Figure 3.2. The result documents by each search method with the keyword “printer” ............ 43
Figure 3.3. An optimising process of a rule tree ..................................................................... 44

Figure 4.1. The concept lattice of the formal context in Table 4.1 ............................................ 55
Figure 4.2. A scale context for the attribute price (S_price) in Table 4.4 and its concept lattice ... 60
Figure 4.3. A scale context for the attribute transmission (S_trans) and its concept lattice ...... 60
Figure 4.4. Concept lattice for the derived context in Table 4.5 ............................................. 61
Figure 4.5. Combined scales for price and transmission using a nested line diagram .......... 62
Figure 4.6. An example of a line diagram (extracted from Groh et al. 1998) ....................... 67

Figure 5.1. A concept lattice of the formal context C in Table 5.1 ........................................... 75
Figure 5.2. The annotating process of keywords for a document .......................................... 82
Figure 5.3. Examples of hierarchies extracted from taxonomies ........................................... 85
Figure 5.4. A lattice £(D’, K’, Γ) of the formal context C’ from Figure 5.1 ...................... 90
Figure 5.5. An example of a lattice structure ......................................................................... 95
Figure 5.6. Partially ordered multi-valued attributes for the domain of research interests ...... 101
Figure 5.7. Examples of nested structures corresponding to concepts .................................. 102
Figure 5.8. An example of pop-up and pull-down menus for the nested structure of a concept103
Figure 5.9. A conceptual scale for the grouping name “databases” .................................... 105

Figure 6.1. Architecture of the system .................................................................................... 110
Figure 6.2. An example of a browsing structure .................................................................... 112
Figure 6.3. An example for the annotation of a home page .................................................. 115
Figure 6.4. An example of selecting topics from other researchers ....................................... 116
Figure 6.5. An example of displaying possible relevant topics for the page being annotated ... 117
Figure 6.6. An example of relevant pages with the page being annotated ............................ 119
Figure 6.7. An example of identifying related pages ............................................................ 120
Figure 6.8. An example of adding new terms ....................................................................... 121
Figure 6.9. An example of editing grouping names ............................................................... 123
Figure 6.10. A snapshot of browsing the top-level concepts .................................................. 124
Figure 6.11. An example of a browsing structure .................................................................. 125
Figure 6.12. An example of the main features of the lattice browsing interface ..................... 126
Figure 6.13. An example of a textword search ....................................................................... 127
Figure 6.14. An example of the nested structure of a concept .............................................. 128
Figure 6.15. The search result with the selection of nested items ........................................... 129
Figure 6.16. An example of the search result extended by a taxonomy ................................. 130
Figure 6.17. An example of a search result and a hierarchical clustering .............................. 132
Figure 6.18. An example of navigating the concept lattice ..................................................... 133
Figure 6.19. An example of a nested structure for a grouping ............................................. 134
Figure 7.1. Questionnaire used for the annotation mechanisms .................................................. 144
Figure 7.2. An example of a different view on the hierarchies of terms ........................................ 153
Figure 7.3(a). Examples of the browsing structure that evolved ................................................. 156
Figure 7.3(b). Examples of the browsing structure that evolved .................................................. 157
Figure 7.3(c). Examples of the browsing structure that evolved ................................................. 158
Figure 7.4. The first and second questions used in the survey of lattice-based browsing .......... 161
Figure 7.5. The third and forth questions used in the survey of lattice-based browsing .............. 162
Figure 7.6. The questionnaire results on “What did you find?” .................................................... 166
List of Tables

Table 4.1. Formal context for a part of “the Animal Kingdom” .............................................. 52
Table 4.2. A procedure of finding formal concepts from the context in Table 4.1 ...................... 54
Table 4.3. Summary for the time complexity and polynomial delay of algorithms .............. 57
Table 4.4. An example of a many-valued context for a part of a “used car market” ............ 59
Table 4.5. A realised scale context for the scale price in Figure 4.2 .................................... 61

Table 5.1. A part of the formal context in the proposed system ............................................ 74
Table 5.2. An example of the many-valued context for the domain of research interests .... 100
Table 5.3. Examples of groupings for scales in the one-valued context ............................... 104

Table 7.1. Number of pages annotated .................................................................................. 138
Table 7.2. Task for each phase of the annotation process .................................................. 139
Table 7.3. Number of terms added at each phase for 59 home pages ................................. 139
Table 7.4. Examples of abbreviation classes registered to the system ................................ 142
Table 7.5. The questionnaire results on the annotation mechanisms ................................... 145
Table 7.6. The questionnaire results on the research topics supported ................................. 146
Table 7.7. Cross-distribution between the number of topics on the list and their generality ... 147
Table 7.8. Cross-distribution between the number of topics on the list and their appropriateness .............................................................. .................................. 148
Table 7.9. Cross-distribution between appropriateness and helpfulness of the listed topics ... 148
Table 7.10. The percentage of the selected terms among the relevant taxonomy terms ......... 150
Table 7.11. Document retrieval using various taxonomies ................................................... 152
Table 7.12. The respondents’ information .......................................................................... 163
Table 7.13. The purpose of the use of the system ................................................................. 164
Table 7.14. The questionnaire results on retrieval performance .......................................... 165
Table 7.15. A cross table with respondents and the reasons they failed for retrieval .......... 165
Table 7.16. Cross-distribution between the used search methods and the number of steps taken .............................................................. .................................. 168
Table 7.17. User opinion on search methods for domain-specific document retrieval ....... 169
Table 7.18. Cross-distribution between lattice-based and hierarchical browsing choices .... 170
Table 7.19. Cross-distribution between lattice-based browsing and Boolean query choices .. 170
Table 7.20. The questionnaire results on the system performance and the user interface ...... 171
Chapter 1
Introduction

1.1. Document Management and Retrieval for Specialised Domains

The World Wide Web is taking over as the main means of providing information. Keeping pace with this evolution of the Web, there is a great demand from organisations as well as individuals for document management and retrieval systems for specialised domains on the Web. Better organisation of documents can make it easier for users to readily find the information they want.

There are many search applications that are feature-packed and high-end commercial products based on conventional information retrieval mechanisms such as Alta Vista, Infoseek and Excite (Sullivan 2000; Platt 1998). These softwares can be used to index the information on local Web sites. Current document management and retrieval systems of organisations greatly depend on such retrieval applications. Extraordinary progress has been made to the point that the general search engines are used for innumerable requests for finding information on the Web.

Despite improvements in this area (e.g., Google and Teoma), specific queries for getting information remain very frustrating. The only search terms the user can think of, may occur in a myriad of other contexts and perhaps do not even occur in some relevant documents. The obvious problem with general search systems is in finding the particular documents that are relevant to one’s interest, query or particular task of the moment. Another major problem of using information retrieval is the difficulty of finding or setting appropriate keywords when one fails to get a search result (Rousseau et al. 1998). In addition, general search engines make no use of domain knowledge and force users to look at a linear display of loosely organised search results.
Some search systems support a better browsing interface (e.g., Alta Vista, Yahoo and Open Directory Projects) using a handcrafted organisation of documents. However, such systems are costly to build and maintain. More recently, dynamic clustering search engines (e.g., Vvisimo and WiseNut) have emerged with an automatic document clustering feature. These systems also make no use of domain knowledge emphasising only syntactical analysis between queries and documents as general search engine mechanisms. The inability to fully analyse the content of a document semantically can result in low precision.

In view of this, general retrieval mechanisms may not always be the ideal tool to use when trying to find specific information, particularly for specialised-domains. There is a need to develop a new approach for domain-specific search mechanisms, rather than simply using conventional document retrieval systems. The new mechanism should manage documents for specialised domains of organisations so that they can be readily and precisely retrieved. It should be easy to build for a given set of documents and be able to deal with changes easily.

In response to the problems of general search engines, there are new research initiatives such as the Semantic Web community portal\(^1\) and the W3C Web Ontology Working group\(^2\), which have the goal of enriching the information in documents for better retrieval. The biggest emerging research area is the use of ontologies to explore the potential of associating Web content with explicit meaning\(^3\). Underlying these research initiatives is the belief that to make full use of the resources on the World Wide Web, documents would have to be marked up according to agreed ontological standards. This means that improved search and better organisation of documents will only be possible by encoding machine processable semantics in the context of the documents using ontologies. The HyperText Markup Language (HTML) does not support specifying


\(^{3}\) For the discussion here an ontology is simply an agreed naming and description convention for the domain.
semantics, but only formatting and hyperlinks. As a consequence, the research initiatives have developed new representation techniques for documents to encode semantics based on ontology representation schemes such as XML/S\textsuperscript{4}, RDF/S\textsuperscript{5}, OIL\textsuperscript{6}, DAML\textsuperscript{7} and DAML+OIL. Some semantic searches are already available on the Web\textsuperscript{8} for special purposes.

In general, this approach assumes knowledge engineers or ontology developers will build ontologies first for a specific subject domain. It then requires users to gain some mastery of the particular ontology to annotate documents to fit into the ontologies, or uses specialists in the ontologies to do the annotation. There is also research into automatic mark-up, but this is a longer term goal. Ontology-based retrieval is intended to allow users to access information more accurately and explicitly. There are likely to be considerable practical advantages to even very large communities committing to specific ontologies, and part of the education process would be to learn the relevant ontologies. For knowledge management on the Web - facilitating knowledge sharing and reuse - the contribution of the ontological approach deserves attention. Moreover, ontology-based reasoning can power advanced information access and navigation by deriving new concepts automatically based on implied inter-ontology relationships.

Despite the practical advantages of a community committing to an ontology, there is also a view that any knowledge structure is a construct. Clancey (1993a; 1997) suggested that when experts are asked to indicate how they solve a problem, they construct an answer rather than recall their problem solving method. There has been a wide range of philosophical discussion on this topic, broadly known as situated cognition. According to Peirce (1931) “knowledge is always under construction,

\textsuperscript{6} OIL (Ontology Inference Layer): http://www.ontoknowledge.org/oil (2002).
\textsuperscript{7} DAML (DARPA Agent Markup Language): http://www.daml.org (2002).
incomplete and continuously assured by human discourse within an intersubjective community of communication”. Knowledge acquisition systems which try to take account of the constructed nature of knowledge include Personal Construct Psychology (Gaines and Shaw 1990) and Ripple-Down Rules (Compton and Jansen 1990). Ripple-Down Rules in particular emphasises the evolutionary and changing nature of knowledge. Based on this philosophical perspective, we would like to explore a new approach for a Web-based document management and retrieval system for specialised domains.

1.2. The Aim of this Thesis

An alternative approach for specialised domains or specific Web sites may be to allow users to create their own organisation of documents and to assist them in ensuring improvement of the system’s performance as it evolves. The aim of this thesis is to develop a system suitable for organisations as well as individuals to incrementally and easily build and maintain their own document management system. It should allow free annotation of documents by multiple users and should continue to evolve both in the structure of browsing and in the retrieval performance. This means that the browsing scheme for document retrieval should evolve as the users annotate their documents.

Another aim is to explore the possibilities of document management systems that do not commit to a priori ontologies and expect that all documents will be annotated according to the ontologies. The aim is systems, which should support the users in annotating a document however they like and the ontology will evolve accordingly. Rather than this being totally ad hoc, we would like the system to assist the users to make extensions to the developing ontology which in some way are improvements. However, this approach does not preclude the inclusion of ontologies imported from elsewhere, but only as a resource that the user is free to use, partially or fully.

The system proposed here uses the lattice-based browsing structure supported by the concept lattice of Formal Concept Analysis (Wille 1982). Lattice-based browsing is automatically and incrementally constructed, and is used as the basic structure for retrieval processes. To incrementally improve the search performance of the system,
knowledge acquisition techniques are developed by reusing terms used by others and terms imported from other taxonomies. The goal of incremental development is similar to both Ripple-Down Rules (Compton and Jansen 1990) and Repertory Grids (Gaines and Shaw 1990). The key strategy of the proposed system is to incorporate the advantages of the concept lattice of Formal Concept Analysis appropriate for browsing, while keeping the incremental aspects of Ripple-Down Rules.

In summary, the aim of this thesis is to develop a Web-based document management system for fairly small communities in specialised domains supporting incremental development of the system over time. The system is based on free annotation of documents by users and assists the users in ensuring improvement of the system’s performance as it evolves. The browsing structure is collaboratively created and maintained over time by multiple users as an ontology (taxonomy) develops. The main focus is an emphasis on incremental development and evolution of the system.

To evaluate the proposed approach, experiments were conducted in the application domain of annotating of researchers’ home pages according to their research interests in the School of Computer Science and Engineering, University of New South Wales. There are around 150 research staff and students in the School who generally have home pages indicating their research interests. The aim was to allow staff and students to freely annotate their pages so that they can be found appropriately within an evolving lattice of research topics. The goal was a system to assist prospective students and potential collaborators in finding research relevant to their interests.

We have also set up a system that allows users to annotate papers from the on-line Banff Knowledge Acquisition Proceedings. The aim of this system was to provide some comparability with the ontological approaches such as the KA² initiative.

9 http://pokey.cse.unsw.edu.au/servlets/RI.
1.3. The Structure of this Thesis

Chapter 2 discusses the current state of document management for information retrieval. Firstly, a review of the current general Web search engines including document clustering is carried out. Secondly, we review ontological approaches that aim to better organise documents to support not only better search results but also better reasoning with documents. Thirdly, information retrieval based on Formal Concept Analysis (FCA) is briefly introduced as the core technique of the proposed system is based on FCA. Finally, the proposed approach for a domain-specific document management and retrieval system is briefly outlined.

The first attempt at incremental development of document management systems in this study was based on the techniques of Ripple-Down Rules (RDR). Thus, Chapter 3 provides an overview of RDR with its background and basics, including its strengths and limitations. Secondly, an automatic help desk system where RDR was used for document management and retrieval is presented. Then, the issues relevant to the RDR help desk system are addressed.

Chapter 4 introduces the basic idea of Formal Concept Analysis (FCA) including formal contexts, formal concepts, concept lattices and conceptual scaling. A number of algorithms in the literature for constructing a lattice are also presented. Here lattice-based models for information retrieval where FCA has been applied are reviewed in detail.

Chapter 5 presents a theoretical framework for the Web-based domain-specific document management and retrieval system that we propose. Firstly, basic notions of the proposed system are defined and an incremental algorithm we have developed for building a concept lattice is provided. Secondly, annotation mechanisms, which cooperate with the knowledge acquisition mechanisms as a way of document management, are presented. Thirdly, lattice-based document retrieval both by browsing a concept lattice and using a Boolean query interface is described. Finally, conceptual scaling to associate a lattice browsing structure with an ontological structure is presented.
Chapter 6 describes systems implemented on the World Wide Web to demonstrate the value of the proposed approach. The first system is for obtaining research interests in the School of Computer Science and Engineering, University of New South Wales (UNSW). The second is a system that gives access to the on-line Banff Knowledge Acquisition Proceedings papers.

Chapter 7 presents the experimental results of using the system to find staff and student home pages for research interests at the School of Computer Science and Engineering, UNSW. For the experiment the system was made available on the School Web site and all users’ activities both for searching and annotating their home pages were recorded.

Finally, Chapter 8 provides a brief summary of the thesis. We then conclude with outlining possible directions for further development of the research presented in this thesis.
Chapter 2

Document Management and Retrieval in Literature

This chapter presents the current state of research into document management for information retrieval. The ultimate objective of document management is to organise documents in a better way so users can easily search for the information they want.

The World Wide Web plays a role as a means of organising documents for retrieval. A HyperText Markup Language (HTML) is currently the basic representation language for documents on the Web. Documents are presented in the HTML format and managed for retrieval using a variety of information retrieval techniques. HTML is essentially a text stream with special codes embedded. These codes are a standard protocol for presentation on the screen of a Web browser, rather than encoding machine processable semantic information. Information is described in natural languages in HTML documents. This simplicity has made possible its dramatic success within a short period. Moreover, lay persons without any computer background knowledge can create HTML documents.

But this simplicity also limits its further growth. As indicated earlier, there has been extraordinary progress made in the development of general Web search engines that are able to access the stored information on the Web. Despite improvements in this area, specific queries for getting information still remain very frustrating. Most problems with the current search engines are due to the limitations of natural language processing. The complete extraction of semantic meaning that the authors embed in natural languages is still impracticable.

In response to this problem, many new research initiatives have been set up to enrich Web resources by developing new representation techniques such as XML(S), RDF(S), OIL, DAML and DAML+OIL. This is the emerging Semantic Web aiming to encode
machine processable semantics based on these representation techniques. Here, ontologies play the role of the backbone of the Semantic Web (next version of the Web). These research initiatives believe that to make full use of the resources on the World Wide Web, documents have to mark up according to agreed ontological standards to support more accurate information. There is a great expectation for the value of this approach: “We are going to build a brain of and for humankind” (Fensel and Musen 2001, pp. 25).

This chapter reviews both general Web search mechanisms and ontology-based retrieval, starting with the current general Web search engines including document clustering in Section 2.1. There is a variety of research on information retrieval systems. The research includes developing better statistical mechanisms, indexing techniques, stemming, and clustering algorithms, supporting more diverse search options and logical operations. It also works at improving the user interface dimension and user feedback, visualisation of information, personalisation and natural language processing. The focus of this review is to present the advantages and disadvantages of typical search engine mechanisms in general, rather than looking at every one of these techniques. In Section 2.2, ontological approaches that aim to better organise documents for better search results are presented, including addressing the issues with ontologies from the perspective of the Semantic Web. In Section 2.3, information retrieval based on Formal Concept Analysis (Wille 1982) is briefly introduced, as the proposed approach is based on Formal Concept Analysis. Finally, a brief outline of the proposed approach for a Web document management system for retrieval is presented in Section 2.4.

2.1. General Approach

A Web search engine is a software program that takes a search query from a user, and finds information corresponding to the user’s query from numerous servers on the Internet (He 1998). Each search engine has knowledge of the Web and attempts to provide the required information in response to a user’s information needs. Web searching can be considered as another form of information retrieval, because most of the techniques used in current Web search engines are drawn from Information Retrieval. Web searching deals with semi-structured data (HTML/XML) and a dynamic
collection of documents, whereas Information Retrieval deals with unstructured data (plain text) and generally a static collection of documents.

Search engines collect information on Web pages using Web crawlers (robots) or by user submission. Then, search systems normalise words contained in a collection of documents using automatic algorithms, and index the documents based on the normalised words usually using a vector space model (Salton and McGill 1983) or a probabilistic model (Turtle and Croft 1991). Information is recollected and reindexed at regular intervals.

The major research issues in this area are (1) document representation, (2) query representation, and (3) retrieval methods. Document representation aims to represent the set of documents by capturing the “essences” for fast checking. Query representation provides various options to assist users in obtaining better results. The retrieval finds all documents that are similar to the query and constructs a list of results according to their apparent relevance. Search engines differ primarily in how they handle each of these issues. The ultimate objective of this area is to improve the efficiency and effectiveness of search engines so that they discriminate relevant documents associated with a query from all other documents in the database.

Broadly speaking there are two ways in which a user interacts with search engines (document retrieval systems). In one, the user formulates a specific query and some documents are retrieved in response. In the second approach, the documents are grouped and the document groups are organised into a structure that can be browsed (document clustering). The user searches for documents by navigating this organised structure.

**2.1.1. Boolean Query**

A search engine usually has an empty input box which allows a user to enter a specific query in a sequence of keywords. The search engine then finds a list of Web sites (URLs) that are relevant to the user’s query through its database. These sites may be ranked according to their relevance to the query. This process is normally iterative in that the user refines the query on the basis of the documents retrieved by each query.
The ideal would be that specific queries would always produce the most relevant documents because the user interface is easy to use and can cover the diverse levels of user knowledge and retrieval skills. As indicated previously, despite improvements in this area (e.g., Google\textsuperscript{12} and Teoma\textsuperscript{13}), finding relevant documents on the Web or even a single site, remains a frustrating task. The only search terms the user can think of, occur in a myriad of other contexts. It is frequently difficult to get a search right despite setting up an apparently specific and appropriate query. The normalised words extracted using a variety of statistical mechanisms do not always concisely represent the meaning of the documents due to the limitations of natural language processing. HTML does not support specifying semantics, only formatting and hyperlinks. In addition, general search engines force the user to look at a linear display of loosely organised results.

As a result, documents are indexed in a classification scheme or Web directory in many information retrieval systems. This is often emphasised as a necessity in Information Retrieval for organising documents (Dewey Decimal System\textsuperscript{14}) and for novice users who do not know precisely what they want or how to get it (Conklin 1987; Landauer et al. 1982; Thompson and Croft 1989; Oddy 1977). With browsing\textsuperscript{15}, users can quickly explore the search domains and can easily acquire domain knowledge (Marchionini and Shneiderman 1988).

The following section presents how documents can be clustered (categorised) for browsing, and reviews current clustering search engines.

\textsuperscript{12} Google (http://www.google.com) is a Web search engine and implements a ranking algorithm based on listing the most popular Web sites first. It is a simple technique based on the assumption that those are most likely to be the sites someone is searching for, improving search performance dramatically.

\textsuperscript{13} Teoma (http://www.teoma.com) is a new search engine similar to Google, but Teoma uses subject-specific popularity, not just general popularity like Google. Subject-specific popularity ranks a site based on a number of same-subject pages that reference it. This means that Teoma generates a list of subjects from the results of a query and identifies the list of subjects. Then, it analyses the relationship of sites within a subject. Teoma also present a set of refinement terms to allow users to clarify their queries. However, it is not yet proven that Teoma produces a better result than Google.

\textsuperscript{14} Dewey Decimal System (http://www.oclc.org/dewey/) is the most widely used library classification system in the world. It has been used for over a century.

\textsuperscript{15} Browsing is a navigation process of given structures to reach the target information or knowledge.
2.1.2. Clustering

The concept of clustering has been investigated for as long as there have been libraries (Kowalski 1997). It has proved an important tool for constructing a taxonomy of a domain by the grouping of closely related documents (Faloutsos and Oard 1995; Frakes and Baeza-Yates 1992; Salton and McGill 1983). Clustering is also used for a classification scheme (Duda and Hart 1973) and has been suggested as a method for formulating browsing (Cutting et al. 1993).

There are two ways in which clustering is constructed with information retrieval systems: pre-clustering and post-clustering. Document clustering has been traditionally examined based on pre-clustering (Van Rijsbergen 1979). In this approach, clustering is performed on all documents in advance and constructs a classification scheme (subject categories). Documents are then located in relation to the subject categories by a similarity measure between the subject terms and the content of documents. Most directory systems used on the Web (Alta Vista, Excite, Yahoo and so on) follow this paradigm. But such manual clustering systems are costly to build and maintain.

Another approach of clustering is based on post-clustering (Croft 1978; Cutting et al. 1992; Leouski and Croft 1996; Charikar et al. 1997; Zamir and Etzioni 1998). In this approach, the clustering is applied on the returned documents corresponding to a user’s query so that it produces more precise results than a pre-clustering approach (Hearst and Pedersen 1996; Zamir and Etzioni 1999). Clustering search engines such as Vivisimo\(^\text{16}\) and WiseNut\(^\text{17}\) follow this paradigm.

These search engines are obviously a huge leap forward in dynamic Web clustering and help users when they are exploring very broad subjects, or when they are looking for something obscure. This is because clustering search engines automatically and

\(^{16}\) Vivisimo (http://vivisimo.com/) is clustering search engine. It analyses the snippets (title, URLs and short descriptions) in the search results of a query, and clusters the results into hierarchical sub-categories. By clicking on a sub-category, a user can get a result page showing only the selected category.

\(^{17}\) WiseNut (http://www.wisenut.com/) is also a clustering search engine similar to Vivisimo, but not nearly as well done as Vivisimo (http://websearch.about.com/library/searchtips/bltotd010905.htm, 2002).
dynamically organise search results of a query into hierarchical sub-categories and these categories can make it easier for the users to refine their query.

However, the efficiency of search performance is still in question, because those approaches also make no use of domain knowledge, emphasising only syntactical analysis between queries and documents. The words in documents or snippets do not always represent the meaning of the documents. Some may be relevant, but others will not be. In addition, the words which represent the meaning of the documents do not always exist in the content of the documents. Moreover, most clustering only focuses on grouping closely related documents into the same cluster (class) and building a one or two level hierarchical tree structure in which each cluster has exactly one parent. That is, the clustering only formulates relationships between parent and child classes, but does not formulate the relationship between classes in the different branches of the hierarchy. This can cause the problem of category mismatch\(^{18}\) (Furnas et al. 1983) where one wrong decision can be critical in failing to find the right documents, and contributes to the low performance of these techniques. If one goes down the wrong path one must go back up the hierarchy and start again. There is no mechanism for navigating to other clusters, as there is only a simple taxonomy structure.

### 2.2. Ontological Approach

In response to the problems of general search engines, new research initiatives have been set up to enrich Web resources to allow better retrieval. Many researchers consider that full use of the Web is only possible by encoding machine processable semantics in the content of the Web presented in the HTML format. Here, ontologies play the role of the semantics for the Web resources. In this approach, ontologies are built first for a domain or a specific subject area, and documents in the domain are annotated based on the ontologies. Then, ontology-based retrieval is supported based on the annotated ontologies to enable more accurate searches. This allows a simple Boolean search to extend to complex higher-order searches. For example, “find all companies which had a profit increase in 2002 that was less than its profit increase in 2001”. The structures of

\(^{18}\) A category mismatch is a violation of the default correspondence between categories at different levels of representation.
ontologies are also utilised as a browsing scheme. One of the main aims of this approach is to facilitate the sharing of information between communities as well as individuals within the groups.

The biggest emerging research area with ontologies is the Semantic Web\(^\text{19}\) for exploring the potential of associating Web content with explicit meaning. Figure 2.1 shows an ontology infrastructure for the Semantic Web. It evolves with Web-based ontology representation languages such as XML/S, RDF/S, OIL, DAML and DAML+OIL. The Web Ontology Working Group\(^\text{20}\) has also been founded by the W3C consortium to construct a standard Ontology Web Language (OWL) for the emerging Semantic Web. One hundred and ninety four ontologies covering a wide range of topics are available on the DARPA Web site\(^\text{21}\). An ontology-based search is also available for DAML annotated Web pages (Li et al. 2002)\(^\text{22}\).

\[\text{Figure 2.1. An infrastructure of the Semantic Web.}\]

\(^{19}\)“The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation” (Berners-Lee et al. 2001). In other words, the Semantic Web (http://www.semanticweb.org) is a vision for the future of the Web to power more explicit Web search by sharing and integrating information available on the Web. Ontologies are the backbone of the Semantic Web.

\(^{20}\)http://www.w3.org/TR/weboant-req (work in progress, 2002).

\(^{21}\)http://www.daml.org/ontologies/ (work in progress, 2002).

A more specific example of this type of activity is the KA\textsuperscript{2} initiative\textsuperscript{23} (Benjamins and Fensel 1998; Benjamins et al. 1999; Staab et al. 2000). KA\textsuperscript{2} starts out with ontologies appropriate to the domain of knowledge acquisition with the expectation that people in the community will annotate documents according to those ontologies. These same users should also be able to use the ontologies to retrieve documents entered by others, or to use the structures of the ontologies for browsing. The KA\textsuperscript{2} initiative has 8-subontologies: organisations, projects, persons, research-topics, publications, events, research-products and research-groups. Each ontology has its own classes, sub-classes, attributes, values and relations. There are some hierarchical relationships between classes and sub-classes. The ontologies are described with the ontology representation language OIL\textsuperscript{24} and DAML+OIL\textsuperscript{25}. Figure 2.2 shows an example of the annotated page of a researcher based on the ontologies\textsuperscript{26}. A demonstration system is also available at the Web site: http://ka2portal.aifb.uni-karlsruhe.de/. The system supports knowledge retrieval to access more accurate information based on the annotated pages for the knowledge acquisition community.

There seem to be many potential benefits from ontologies in performing high quality semantic searches. Ontology-based retrieval can empower advanced information access and navigation by deriving new concepts automatically based on implied inter-ontology relationships (automated reasoning). Users will be able to conduct more accurate searches, and to find and learn more than they expected. For example, suppose that a user is looking for the address of a certain person (here “Steffen Staab” shown in Figure 2.2) using the ontological browser of KA\textsuperscript{2}. The system may present more information than the user expected such as the person’s affiliation, e-mail, research interests, projects which were annotated based on the ontologies, as shown in Figure 2.3.

\textsuperscript{23} This research aims at intelligent knowledge retrieval from the Web. Another objective of the initiative is to gain better insight in distributive ontological engineering processes. The researches choose “the knowledge acquisition community” as an ontology domain to model.


\textsuperscript{26} This extracts from the Web site (http://www.aifb.uni-karlsruhe.de/~sst/) by viewing the source code of the page (2002).
Figure 2.2. An instantiation example of the ontologies (an annotated home page).
However, it still remains an unproven conjecture that ontological approaches will enhance search capabilities (Uschold 2002). Semantic querying capabilities are active areas of research, but the computational properties of such a query language, both theoretical and empirical, are yet to be determined (Horrocks 2002). There are also a number of critical issues relating to ontological approaches that need to be addressed. These issues are discussed in Section 2.2.3.

| affiliation | http://www.aifb.uni-karlsruhe.de/ | Institute |
| email | mailto: stefen@karlsruhe.de | STRING |
| firstName | Stef en | STRING |
| lastName | Stef en | STRING |
| name | Stef en Staab | STRING |
| researchInterest | Knowledge Management | ResearchTopic |
| researchInterest | Ontology Engineering | ResearchTopic |
| worksAtProject | http://www.aifb.uni-karlsruhe.de | Project |
| worksAtProject | http://www.aifb.uni-karlsruhe.de/WBS/broker | Project |
| address | Institute for Applied Computer Science and Formal Description Methods (ALFB), Universitat Karlsruhe (TH), Karlsruhe, Germany | STRING |
| affiliation | http://www.aifb.uni-karlsruhe.de | Institute |
| email | mailto: studer@aifb.uni-karlsruhe.de | STRING |
| fax | ++49-(0)721-6937177 | STRING |
| firstName | Studer | STRING |
| headOfGroup | http://www.aifb.uni-karlsruhe.de/WBS | ResearchGroup |
| lastName | Studer | STRING |
| memberOf | http://www.aifb.uni-karlsruhe.de/WBS | ResearchGroup |
To review ontological approaches for knowledge management and retrieval more fully, the definition of an ontology and what the goals are that people pursue in ontology communities will be examined. Secondly, the types of ontologies based on a standard for categorising ontologies will be discussed. Finally, the issues relevant to ontologies such as ontology construction, the knowledge acquisition bottleneck and the user interface with the evolution of ontologies will be observed.

2.2.1. A Notion of Ontology

Recently, ontology has become a major subject of interest as a powerful way to express the nature of a domain. In the knowledge engineering community, ontologies have also become popular due to the growing importance of knowledge integration, sharing and reuse in a formal and task independent way. What ontologies are is still a debated issue. Various definitions of ontology have been presented in the literature. The most cited definition of an ontology in the knowledge engineering community is as follows: An ontology is an explicit specification of a conceptualisation (Gruber 1993).

A conceptualisation is an abstract and simplified view of the world. It is a process to identify a set of abstract objects, concepts and other entities which are presumed to exist in a certain domain and the relationships that hold among them (Genesereth and Nilsson 1987). Any real world situation can be considered as a particular instantiation of an ontology. In addition, any knowledge-based system requires some representation of the world over which it reasons. A central part of knowledge representation for a domain (a part of the world) is based on elaborating a conceptualisation (Valente and Breuker 1996) and building an ontology. Elaborating a conceptualisation is an essential component for knowledge representation tasks, because conceptualisations abstract which things are relevant to be represented and which are not (Davis et al. 1993).

Guarino (1997) cited and reviewed a number of definitions trying to establish a comprehensive definition of an ontology. In a later paper, he refines Gruber’s definition (1993) by making clear the difference between an ontology and a conceptualisation as follows:
“An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models” (Guarino 1998, pp. 7).

Most researchers agree that an ontology must include a vocabulary and its definitions, even where there is no consensus on a more detailed characterisation and the definitions, are often vague (Heflin 2001). Typically, a formal ontology consists of terms, definitions and formal axioms relating them together (Gruber 1993). The definitions associate the names of entities in the world such as classes, relations, functions, and constraints.

Guarino (1995) underlined the necessity for formal ontological principles based on the interdisciplinary perspective within the knowledge engineering community. First of all, he pointed out the principles of formal ontologies based on the modelling view of knowledge acquisition proposed by Clancey, “the primary concern of knowledge engineering is modelling systems in the world, not replicating how people think” (Clancey 1993b, pp.34). In other words, a knowledge base must be the result of a modelling activity relating to an external environment, rather than a repository of knowledge extracted from an expert’s mind. Gaines (1993), Gruber (1995) and Schreiber et al. (1993) sustain a similar view.

Following this perspective, formal ontologies aim to make conceptual modelling less dependent on particular perspectives. Another principle of formal ontologies is to facilitate a communication between diverse communities. Additionally, ontologies support knowledge sharing (Musen 1992; Gruber 1993; Gruber 1995; Pirlein and Studer 1995). Ontologies can share and reuse other ontologies or at least parts of them, for a variety of different purposes. If there is a well-developed ontology, another ontology can use the first without having to remodel it.
Sharing various underlying definitions, ontologies can be distinguished into different types. General types of ontologies are described followed by a view of (Laresgoiti et al. 1996; Studer et al. 1998; van Heijst et al. 1997).

2.2.2. Types of Ontologies

Ontologies can be identified under four major categories, namely generic ontologies, representation ontologies, domain ontologies, and application ontologies, depending on their generalisation levels or the subject of the conceptualisation. Each ontology category is briefly described using an example. This however, is not a standard for categorising ontologies. There are also other ways of describing ontologies such as information ontology, enterprise ontology, method ontology, upper-level ontology, lower-level ontology, taxonomical ontology and others.

Generic Ontologies

General ontologies are also referred to as upper-level ontologies or as core ontologies (van Heijst et al. 1997). These ontologies usually represent general world knowledge. In the upper-level ontologies, a taxonomy tends to be the central part of the ontologies. Terms in the world are typically organised in a taxonomy, even when there is some disagreement about the hierarchy among ontology researchers. All upper-level ontologies try to categorise the same world, but they are very different at their top-level (Noy and Hafner 1997).

Cyc\(^{27}\) and Sowa’s ontology (2000) can be considered as typical generic ontologies. WordNet\(^{28}\), one of the most well developed lexical ontologies, can also be classified in this category. Figure 2.4 shows the top level of the Cyc hierarchy.

\(^{27}\) To create a general ontology for commonsense knowledge, Cyc (http://www.cyc.com/) was founded by Doug Lenat in 1994 (Lenat 1995, Lenat and Guha 1990, also see the Web site: http://www.cyc.com/cyc-2-1/cover.html). The knowledge base is built upon a core of over 1,000,000 hand-entered assertions (or “rules”) designed to capture a large portion of consensus knowledge about the world.

Figure 2.4. Top-level categories of Cyc (adapted from Lenat and Guha 1990).

**Representation Ontologies**

Such ontologies provide a representational framework without committing to any particular domain. An example of this category is the *Frame ontology* (Gruber 1993), which allows users to define concepts of an evolved domain (frames, slots, relations and constraints of the slots). Users can build a knowledge base by instantiating the concepts they define. Ontolingua’s *Frame Ontology*\(^{29}\) is the most representative ontology and for some years has been considered as a standard in the ontology community.

**Domain Ontologies**

Domain ontologies specify the knowledge for a particular type of domain. Such ontologies generalise over application tasks in that domain; such as medical, electronic or other domains. The KA\(^2\) initiative (Benjamins et al. 1999; Staab et al. 2000) can be categorised as this type of ontology. Ontologies to facilitate the Semantic Web can be also categorised as these domain ontologies.

\(^{29}\) Ontolingua is the ontology building language used by the Ontolingua Server (Farquhar at al. 1997, also see the Web site: [http://www-ksl-svc.stanford.edu:5915/](http://www-ksl-svc.stanford.edu:5915/)).
Application Ontologies

An application ontology is an ontology used by a particular application containing all the definitions required for knowledge modelling in the application. It also contains the information structures for building an application system. Typically, application ontologies are related to the particular tasks of the application. People can construct an application ontology adapted to a particular task at hand by importing from existing ontologies. That is, an application ontology can be formulated by adapting and merging existing domain and generic ontologies to suit them to a particular task and a domain.

Within each major categorisation of ontologies above, the description levels of the ontologies are diverse. For example, the Open Directory Project is categorised as a generic ontology (one of the world’s biggest taxonomies). But only the definitions of the terms used in this directory system are established. Most application ontologies for enterprise applications simply use the structure of the domain ontologies (classes, subclasses and attributes), even though many are turning to a more formal ontology to accurately share information and interact between communities. The ontology that is defined by the Web-Ontology Working Group\(^{30}\) to facilitate the Semantic Web requires specification of classes, attributes and their relationships. This is much simpler than a formal ontology which is required within ontology communities.

2.2.3. The Issues relevant to Ontologies

Many researchers believe that improved search is only possible by using ontologies to encode machine processable semantics in the content of the documents. There are likely to be considerable practical advantages through using various specialised reasoning services. The explicit representation of the semantics underlying Web pages and resources should enable intelligent access of heterogeneous and distributed knowledge, and a qualitatively better level of service (Ding et al. 2002).

\(^{30}\) The working group \(\text{http://www.w3.org/TR/ont-req, 2002}\) has not reached consensus on all topics as open issues are still under discussion (work in progress). But the current specified requirements for an ontology are its classes (general thing), the relationships that can exist among things and the properties (attributes) those things may have.
Although ontologies promise to solve many knowledge management and retrieval problems on the Web and can play a key role in the Semantic Web, these promises contain many assumptions such as well constructed ontologies, well annotated pages, knowledge annotation mechanisms synchronised with ontology evolution, sophisticated semantic querying capabilities and others.

**Ontology Construction**
To facilitate communication between agents and people based on ontologies, first of all, a standard ontology definition and language are required. There are a number of different ontology representation languages such as RDF, OIL, DAML, DAML+OIL. However, the meaning of the term *ontology* is often vague and still there is no widely accepted formal definition of an ontology, even though communities are trying to specify common consensus ontologies. Communities try to follow the principle of ontologies when they construct ontologies, but to some extent ontologies are highly varied in reality. An application must commit to the same consensus ontologies for shared meaning to be able to access explicit knowledge sanctioned by the ontologies for inference engines. To accommodate this issue, an ontology-working group has been formed to develop a W3C standard ontology language (OWL). However, more flexible mechanisms will be better than enforcing the use of a standard language. Translation mechanisms can be an alternative for compatibility between the existing ontology languages.

Other important issues currently facing ontology research communities relate to ontology evolution, ontology extension and ontology divergence (Ding et al. 2002; Heflin 2001; W3C technical report31 2002). Knowledge is constantly changing so ontologies will change over time. Thus, the management of ontology change is necessary for consistency with the corresponding changes to knowledge and information. A Web ontology language and inference engine must accommodate ontology evolution. One prominent aim of ontologies is to facilitate knowledge sharing and reuse. A large ontology can be developed by combining, adding and refining existing ontologies. To achieve this aim, ontologies must use the same terms and

31 [http://www.w3.org/TR/webont-req](http://www.w3.org/TR/webont-req) (work in progress, 2002).
axioms to model similar concepts and must manage ontology extension. Inference engines also need to refer to the content of the extended ontology concepts. But most current ontology systems do not accommodate extension (Heflin 2001). When an agent needs to develop an application, they can use the existing ontologies, but the ontologies are often not sufficient and are not easily merged with each other. Issues relating to ontology extension concern: how agents can extend existing ontologies, and how inference engines take into account the extended ontology that may be critical for the knowledge organisation. Thus, the ontology must be designed to adapt well and complement other ontologies when considering potential applications.

Ontology communities try to build a standard ontology for a domain, but the existence of diverse ontologies for the same domain is unavoidable. Different people can build different ontologies for the same domain. When an agent builds a domain-specific ontology for an application, the agent can use shared ontologies, but the extension of an existing ontology is often needed. This rule applies to multiple agents building similar applications in the same domain. As a consequence, application ontologies can be diverse for the same domain. Therefore, integration mechanisms will be necessary to accommodate ontology divergence. Two different terminologies can have the same meaning and the same terminology can have different meanings. The four state conditions (consensus, correspondence, conflict and contrast) of Gaines and Shaw (1989) for shared knowledge construction should be considered when different ontologies are integrated.

**Knowledge Acquisition Bottleneck**

One of the major issues relating to ontologies is the annotation of ontologies into the content of documents for machine processible semantics. In theory, automated annotation tools (e.g., AeroDAML\[^{32}\]) may overcome the knowledge acquisition bottleneck. However, due to the limitations of NLP (Natural Language Processing), complete automatic annotation is unrealistic (Heflin 2001). Semi-automatic methods, where human annotators are involved in the annotation process based on techniques

---

[^32]: AeroDAML ([http://ubit.lockheedmartin.com/ubit/hotdaml/aerodaml.html/](http://ubit.lockheedmartin.com/ubit/hotdaml/aerodaml.html/)) is a knowledge markup tool that automatically generates DAML annotation from Web pages (Kogut and Holmes 2001).
from natural language processing, machine learning and information extraction, may be the optimal solution. A number of semi-automatic semantic annotation tools (e.g., OntoAnnotate\textsuperscript{33}, OntoMat\textsuperscript{34}, and SHOE knowledge annotator\textsuperscript{35}) are available. However, ontology evolution can cause inconsistency between the ontologies and the contents of annotated documents or meta-data. Ontologies evolve so the annotation process may need to evolve to synchronise with the corresponding changes to ontologies.

**User Interface**

Another important area is the query interface. By using an ontological browser, users may not need to know complete ontologies. However, the users are somehow required to understand the available ontological terms sufficiently to establish a query able to infer implicated knowledge in ontologies. Examples can be seen in the DARPA and KA\textsuperscript{2} initiative Web sites\textsuperscript{36}. The users may not want to look at the ontological terms and notions just to form a query. They may prefer to find information with just one or two query words at first and then refine their query if they are not satisfied with the search results in a typical retrieval fashion, rather than looking for ontological terms at the first stage. Of course, keyword search is often inadequate and a parametric search can be more useful than a keyword search, but there is a significant issue from the useability point of view in specifying pairs of attributes and values to build a query. Thus, how the context of the ontological query interface can be designed to allow users to learn about the contents of ontologies in order to create the desired query is also a challenge.

A user interface based on ontological structures can be useful when users know exactly what they want to find. On the other hand, if the information one is seeking is not represented in the ontology, or one does not understand the relation of the ontology to one’s query, s/he has the same problem as with general search engines. The ideal approach would be to support a combined mechanism to choose a method simultaneously such as a semantic ontological search, ontological browsing interface,

\textsuperscript{33} http://www.ontoprise.com/ (2002).
\textsuperscript{34} http://annotation.semanticweb.org/ontomat/index.html (2002).
and typical retrieval interface which uses Boolean query and browsing of subject categories. Even though the ontological approach can allow users to access explicit and exact information by browsing the structures of ontologies, the user may still require a search by Boolean querying or classification retrieval used in general search engines.

Note that not all issues relating to ontological approaches are discussed in this section. Knowledge inconsistency encoded in resources, scalability to the Web, ontology interoperability and ontology learning are also important issues relating to ontology approaches. Some issues described above are relevant to the realisation of the Semantic Web vision, rather than domain-specific knowledge management and retrieval that is the aim of this thesis.

The study of ontologies deals with the a priori nature of reality to capture universally valid knowledge (Guarino 1995). We believe that most of ontology issues arise from this belief. However, in spite of these issues, there seems to be many potential benefits from ontologies in facilitating the sharing of knowledge between and within communities, as well as in performing high quality semantic searches.

Despite the practical advantages of a community committing to ontologies, there is also a view that any knowledge structure is a construct, which should be allowed to evolve over time (Compton and Jansen 1990) as indicated in Chapter 1. Peirce (1931) noted that knowledge is always under construction and incomplete. Situated cognition suggests that when experts are asked to indicate how they solve a problem, they construct an answer rather than recall their problem solving method (Clancey 1993a). Personal Construct Psychology (Gaines and Shaw 1990) and Ripple-Down Rules (Compton and Jansen 1990) also account for the constructed nature of knowledge. We can also expect that increasingly the interactions in which explicit knowledge emerges will only arise during an interactive and iterative communication involving some sort of system (Stumme et al.1998). Based on this philosophical background, we would like to explore a new approach for a Web-based domain-specific document management and retrieval system. This approach focuses on incremental construction of knowledge in the context of its use based on the situated cognition view.
Rather than committing to \textit{a priori} ontologies and expecting that all documents will be annotated according to the ontologies, the aim of this thesis is to explore the possibilities of a system where a user can annotate a document however they like and that the ontologies emerge from this. Rather than this being totally \textit{ad hoc}, we would like the system to assist the user to make extensions to the emerging ontologies that are improvements. We are not concerned with automated or semi-automated ways of discovering an ontology appropriate to a document or corpus (Aussenac-Gilles et al. 2000; Maedche and Staab 2000). Despite the potential of such approaches, from our more deconstructionist perspective, we are more interested in the role of the reader or user interpreting documents and deciding on their annotation and the development of an ontology. The user here may be the individual user, an expert for a specialised domain or a small community.

However, this does not preclude the inclusion of ontologies either constructed by an expert or ontologies imported from elsewhere, as part of the ontological structure preferred by the user. We do not propose a completely \textit{ad hoc} evolution of an ontology. It is perfectly sensible for the individual user or group to be influenced by existing ontological standards, and interfaces should support this. However, rather than being locked into conforming to a standard, the user should be free to use all, small fragments, or none of the ontology as best suits their purpose. A new ontology will emerge as their result, and this itself may become a useful ontology for other groups.

\subsection*{2.3. Formal Concept Analysis Approach}

Another approach is based on lattice-based information retrieval using Formal Concept Analysis (FCA - Wille 1982). This has not yet been widely applied to information retrieval. In this approach, documents are annotated with a set of controlled terms by experts or automatic algorithms. Then, using FCA the documents are indexed into a lattice structure that can be used for browsing. In FCA, a concept is specified by an extension as well as intension. The extension of a concept is formed by all objects to which the concept applies and the intension consists of all attributes existing in those objects. These concepts form a lattice structure, where each node is specified by a set of objects and the attributes they share. As one progresses down the lattice more attributes
are added and so each node covers fewer objects. The lattice can be quite sparse and have a range of structures, as a node is added only where the attributes at the node distinguish the objects from those at another node. The mathematics for this are well established and FCA has been successfully applied to a wide range of applications in medicine, psychology, libraries, software engineering and ecology, and to a variety of methods for data analysis, information retrieval, and knowledge discovery in databases.

A number of researchers have advanced this lattice structure for document retrieval (Godin et al. 1993; Carpineto and Romano 1996a; Carpineto and Romano 1996b; Priss 2000b). Several researchers have also studied the lattice-based information retrieval with graphically represented lattices for specific domains such as libraries, flight information, e-mail management and real-estate advertisements (Rock and Wille 2000; Eklund et al. 2000; Cole and Stumme 2000; Cole and Eklund 2001).

The mathematics of Formal Concept Analysis can be considered as a machine learning algorithm which can facilitate automatic document clustering. In other words, FCA cab be considered as one of incremental clustering algorithms based on post-clustering. A key difference between FCA techniques and the general clustering algorithms in IR is that the mathematical formulas of FCA produce a concept lattice which provides all possible generalisation and specialisation relationships between document sets and attribute sets. This means that a concept lattice can represent conceptual hierarchies which are inherent in the data of a particular domain. Thus, the lattice can imply all minimal refinements and minimal enlargements for a query at an edge in the lattice (Godin et al. 1995). This means that following an edge downward corresponds to a minimal refinement for the query at the edge in the lattice, and vice versa. In addition, the hierarchical tree structure, in which each cluster has exactly one parent, can also be embedded into the lattice structure.

Another difference with FCA is in the method of clustering documents. FCA produces a lattice structure for browsing. Each point has multiple parents and children which can be a superior structure to the hierarchical tree. This lattice structure allows one to navigate down to a node by one path, and if a relevant document is not found one can
go back up another path rather than simply starting again. When one navigates down a hierarchy one tries to pick the best child at each step. If the right document is not found it is difficult to know what to do next, because one has already made the best guesses possible at each decision point. However, with a lattice, the ability to go back up via another pathway opens up new decisions, which one has not previously considered.

A more detailed explanation on how the basic theories of FCA are applied to information retrieval will be presented in Chapter 4. The previous work on information retrieval using FCA as well as the differences between the previous work and the proposed system will also be examined.

## 2.4. Proposed Approach

The proposed approach uses Formal Concept Analysis (FCA) for domain-specific document management and retrieval in order to support lattice-based browsing. In other words, the core of the technology in the proposed system is FCA. The difference in the proposed approach is mainly in the way the system is used rather than its underlying FCA basis. The main focus of the proposed system is an emphasis on incremental development and evolution, and knowledge acquisition tools to support these.

The system is aimed at multiple users being able to add and amend document annotations whenever they choose. The users are also assisted in finding appropriate annotations. This results in the automatic generation of a lattice-based browsing system from the terms used for annotations. The users can immediately view the concept lattice that incorporates their documents and further decide whether the terms they assigned for the documents are appropriate. If the browsing does not support the group who annotated the documents, it will be able to rapidly and easily evolve. The browsing structure here is increasingly referred to as an ontology (or taxonomy) which evolves accordingly where users annotate documents in whichever way they like.

The main differences between the previous work on FCA and the proposed system will be presented in Chapter 4. The main features of the proposed system and these details will be presented in Chapter 5.
2.5. Chapter Summary

There has been extraordinary progress in the development of Web retrieval systems improving search performance dramatically. There has also been a huge leap forward in automatic document clustering allowing users to find information faster and helping them especially when they are looking for something obscure.

Recently there has been great interest in having documents conform to ontological standards. The goals that the ontology approach pursues along with its notions and issues were presented. There are likely to be considerable practical advantages through various specialised reasoning services, but overall it remains an unproven conjecture that ontological approaches will enhance search capabilities. There are also critical issues requiring further research to realise a true semantic search.

As an alternative approach, the possibilities of document management systems that do not commit to a priori ontologies was explored. But this does not prevent the inclusion of existing ontologies. Rather this is to explore the possibilities of a system where users can annotate their document in whichever way they like and that ontologies will evolve accordingly. Based on this assumption, an alternative approach based on the lattice-based browsing structure of Formal Concept Analysis was proposed.

The first attempt at incremental development of document management systems in this thesis was based on the Ripple-Down Rules (RDR) techniques so that the next chapter presents the RDR approach with its strengths and limitations.
Chapter 3
Document Management for Retrieval
with Ripple-Down Rules\textsuperscript{37}

Web-based document retrieval systems for a specialised domain (a help desk system) were developed based on the Ripple-Down Rules (RDR) techniques (Kang et al. 1997; Kim et al. 1999). The systems are based on a combination of standard information retrieval techniques and the RDR knowledge acquisition technique. They are the first attempt at incremental development of document management system in this study. This approach to document management has some commercial use in help desk support\textsuperscript{38}. The help system of Kim et al. (1999) allows simple incremental maintenance of the system’s knowledge so that the search performance of the system can be improved over time. The idea behind the system is that when a user fails to find a suitable document, the system would send an expert a log of the interaction. The RDR mechanism then assists the expert to add new keywords so that the correct document will be found next time.

Ripple-Down Rules is an attempt to address knowledge acquisition from a situated cognition perspective (Compton and Jansen 1990). The central idea is that experts are good at creating justifications for why one conclusion should be given rather than another. It has been successfully applied to a range of tasks: knowledge reuse, heuristic search, configuration, machine learning, fuzzy reasoning and others. One of the significant strengths of RDR is that knowledge acquisition and maintenance are simple tasks. With this incremental knowledge acquisition mechanism and robust maintenance methodology, the RDR mechanism has been applied to a help desk system to manage

\textsuperscript{37} This work was developed for a project in a course work master’s degree (Kim 1999) and has been partially reported here. This work followed earlier work (Kang et al. 1997).

\textsuperscript{38} Byeong Kang, personal communication.
help documents. It is essential for a help desk system to have some sort of mechanism such as RDR that allows for easy incremental development and improvement of the system if it fails to deliver in particular situations.

Section 3.1 gives an overview of Ripple-Down Rules with its background and basics including its strengths and limitations. Section 3.2 presents an automated help desk system where RDR was used for document management and retrieval. Finally the issues relevant to the RDR help desk system are discussed.

3.1. Ripple-Down Rules

3.1.1. Background of RDR

A major criticism of the early work on knowledge based systems (KBS) and the traditional software engineering approaches come from a situated cognition perspective which makes the claim that interaction with an expert has been misunderstood. Situated cognition suggests that when experts are asked to indicate how they solve a problem, they construct an answer rather than recall their problem solving method (Clancey 1993a). In particular it seems that they construct an answer to justify that their solution to the problem is appropriate and that this justification depends on the context in which it is asked (Compton and Jansen 1990).

A simple example is that when a clinician is asked why they believe a patient has a certain disease, they will often indicate the symptoms that distinguish the case from other diagnoses the questioner may be considering. This results in a quite different explanation for different questioners and also for the same questioner on different occasions. This results in the maintenance problems that occur with expert systems: that the knowledge provided by an expert is never precise enough or complete enough, even when the knowledge in the domain itself is not developing (Compton et al. 1989). The problem is only exacerbated when, as always occurs, the domain itself is evolving.

To address knowledge acquisition from a situated cognition perspective, Compton and Jansen (Compton and Jansen 1990) invented Ripple-Down Rules (RDR). It is an
effective knowledge acquisition and representation methodology which allows a domain
expert to acquire and maintain knowledge without the help of knowledge engineers.
The original motivation was that experts are good at creating justifications why one
conclusion should be given rather than another (Compton et al. 1989). RDR itself
organises the knowledge, and knowledge acquisition and maintenance are easily
achieved. In the RDR method, the expert is only required to identify features that
differentiate between a new case being added from other stored cases already correctly
handled, without considering the structure of the KB. The emphasis on asking experts
about difference is very similar to the use of differences in Personal Construct
Psychology (Gaines and Shaw 1990).

3.1.2. Basics of RDR
In an RDR framework the task of the expert is to check the output of the developing
KBS. If the expert disagrees with the KBS conclusions, it is because they have
identified some data in the input which suggests an alternative conclusion. The features
or data and the conclusion they suggest can be organised as a rule. However, this rule
was provided in the context of a particular mistake, so the knowledge base is structured
so that this rule is reached only in the same context; that is, if the same sequence of
rules leading to the same mistake is activated again.

There are a number of ways of structuring a KBS in this way to make it suitable for
various tasks. A key feature of any RDR system is that since rules are added because of
cases, any cases that prompt the addition of a rule are stored. None of the stored cases,
which are already handled by the other rules in the system, should cause the new rule to
fire. There are a number of ways to ensure this, but a key method is simply to present a
previous case to the expert that is covered by the rule and ask the expert to select further
features that distinguish the cases. This process is repeated, and even with very large
KBS the expert needs only consider two or three of the stored cases.

RDR systems have been implemented in a wide range of application areas achieving
great success in real world problems. The first industrial demonstration of this approach
was the PEIRS system which provided clinical interpretations for reports of pathology
testing (Edwards et al. 1993). The approach has also been adapted to a range of tasks: multiple classification (Kang et al. 1995), control (Shiraz and Sammut 1997), reuse of knowledge (Richards and Compton 1997b), heuristic search (Beydoun and Hoffmann 1997; 1998a) and configuration (Compton et al. 1998). There are a number of other lines of RDR research integrating RDR with machine learning (Shiraz and Sammut 1998), fuzzy reasoning (Martinez-Bejar, Shiraz et al. 1998) and discovering of ontologies (Suryanto and Compton 2000; 2001).

The first approach of RDR assumed a single conclusion for a case (Single Classification RDR - SCRDR). This produces a decision list with an if-true and if-false structure in a binary tree with a rule at each node. Every node can have two branches to two other rules: one to a true node (an exception branch) and another to a false node. If a case fires a rule, then its child rule (true branch) is evaluated. Otherwise, it is evaluated against its sibling rule (false branch). The conclusion for a case is given from the conclusion of the last satisfied rule in the path to a leaf node.

To extend RDR to multiple conclusions, MCRDR (Multiple Classification RDR) was developed using an n-ary tree (Kang et al. 1995). MCRDR deals with tasks where multiple independent classifications are required. In MCRDR, every rule can only have exception nodes. If a case satisfies a rule, then all its children are evaluated. This process will continue until there are no more child nodes to be evaluated or none of the child rules are satisfied by the case. Conclusions are given from the last satisfied rule in each path.

Fuzzy RDR was developed to model and represent fuzzy domain knowledge for propagating uncertainty values in an RDR knowledge base (Martinez-Bejar, Shiraz et al. 1998; Martinez-Bejar et al. 1999. See also the Fuzzy RDR Web site39).

More recently, RDR was extended to Nested RDR (NRDR) to facilitate incremental acquisition of search knowledge where some attributes are not known a priori (Beydoun and Hoffmann 1997; 1998a; 1999). NRDR uses a single classification RDR structure

and more generally applies to problems. In this structure, a concept is defined by a separate SCRDR tree. The defined concept then can be used to define other concepts. That is, the conditions of a rule in an RDR tree can be provided by input data or by an RDR tree (a concept). When a condition of a rule includes concept(s), the Boolean value of the condition is calculated in a backward chaining mode. The conclusion for a case ends up in one path because NRDR is based on a single classification which is either true or false. Every concept has a dependency list to prevent circularity and recursive definitions. The dependency list is also used to conduct a consistency check in the KB.

An equivalent system is MCRDR with repeat inference which has been used in configuration (Compton et al. 1998) and room allocation (Richard and Compton 1999). This has been generalised (Compton and Richard 1999).

3.1.3. Strengths of RDR

A significant strength of RDR is that knowledge acquisition and maintenance are easily achieved. RDR itself organises the knowledge and the expert is only required to identify features that differentiate between a new case being added and the other stored cases already correctly handled, without considering the structure of the KB.

In the RDR method, a rule is only added to the system when a case has been given a wrong conclusion. Any cases that have prompted knowledge acquisition are stored along with the knowledge base. RDR does not allow the expert to add any rules which would result in any of these stored cases being given different conclusions except by allowing a specific override. This means that the existing rules’ consistency is kept (verification and validation) and that there is incremental improvement in the system.

The level of evaluation in RDR systems varies, but they have invariably shown very simple and highly efficient knowledge acquisition. RDR for the task of providing interpretative comments for Medical Chemical Pathology reports are now available commercially. Results from this experience have not yet been published, but confirm that very large knowledge bases (> 7000 rules) can be built and maintained very easily
by pathologists with little computing experience or knowledge (Pacific Knowledge
Systems, personal communication).

The other critical finding from the RDR evaluations is that this form of knowledge
acquisition results in compact and efficient knowledge bases. It might be expected that
incremental addition of knowledge where knowledge was only added (as a refinement),
but never changed, may result in very large knowledge bases with much repeated
knowledge. However, simulation studies show the size of the knowledge bases are
comparable to those produced by machine learning (Compton et al. 1995; Kang et al.
1998) and there is a significant increase in size only when a random choice is used by
the expert. In studies on a human developed MCRDR knowledge base (~3000) rules,
only 10% compression could be achieved (Suryanto et al. 1999).

3.1.4. Limitations of RDR

Despite great success in a wide range of application areas, the current RDR-based
systems have been criticised for their limitations in supplying an explicit model of the
domain knowledge (Richards and Compton 1997b; Martinez-Bejar, Benjamins et al.
1998; Beydoun and Hoffmann 1997). This means that the RDR methodology does not
support higher-level models, especially abstraction hierarchies. RDR assumes a simple
attribute value representation of the world and supports only rules rather than
inheritance or other deductive reasoning from an ontology.

To address this lack, some work has already been done on evolving hierarchies in
parallel with RDR (ROCH; Martinez-Bejar, Benjamins et al. 1998), discovering
abstraction hierarchies from MCRDR (MCRDR/FCA; Richards and Compton 1997b)
and modelling domain knowledge with simultaneous knowledge acquisition (NRDR:
Beydoun and Hoffmann 1998b). However, more research will be needed to develop the
full potential of RDR based on an ontology concept.

Another limitation of RDR is the repetition within the knowledge base (Beydoun 2000;
Richards 1998). But the repetition problem is not a serious impediment of RDR as
addressed in Suryanto et al. (1999).
3.2. A Help Desk System with Ripple-Down Rules

In many areas, help desk services of various forms are provided to assist users in solving computer related problems. Conventional automated help desk systems (HDS) assist an organisation in automating the help desk process of handling and resolving reported problems.

The World Wide Web today takes over as the main means of providing information and users themselves try to solve their problems by searching for the information through the Web. As a consequence, an automated help desk is necessary to support an information retrieval mechanism to make it easier for users to find what they are looking for. When the users cannot track or solve their requests, they may report their problems to experts who can manage the help desk. The automated HDS also needs to be easily maintained as invariably, there is a dynamic knowledge environment. This means that the help desk system should provide a powerful search and retrieval mechanism as well as a robust maintenance methodology. We undertook a study to develop such a help desk system by applying RDR to HDS. This study extended earlier work where RDR was used for help desk information retrieval and proposed (Kang et al. 1997), but did not actually present incremental maintenance of the system. Most of the theoretical background of this study drew on their work.

An automated help desk is essentially a knowledge-based system because it is related to a user’s problem-solving task. A Case-Based Reasoning (CBR) approach has been proposed as more appropriate to build up help desk systems (Barletta 1993a; Barletta 1993b; Simoudis and Miller 1991). Ripple-Down Rules (RDR) is grounded on a similar philosophy to CBR and can be considered as a system which emphasises both cases and expert knowledge. Previous RDR systems were based on simple attribute value data. In the current work, a case is a document and its keywords: a situation close to the conventional CBR application. Thus, we believed that the RDR methodology could be applied to a help desk system.

---

40 This section is largely taken from the paper “Kim, M., Compton, P. and Kang, B. H. (1999). Incremented Development of a Web Based Help Desk System, Proceedings of the 4th Australian Knowledge Acquisition Workshop (AKAW99), University of NSW, Sydney, 13-29".
3.2.1. Overview of the System

A prototype help system was developed using Multiple Classification RDR (MCRDR) to build and maintain the knowledge base of the system. Help documents were extracted from the Frequently Asked Questions page\textsuperscript{41} maintained by the Help Desk of the School of Computer Science and Engineering, University of New South Wales.

A user can report their problems to a human expert through the system and the expert can refine the knowledge base to deal correctly with the user’s problems. A log of the user session is available for this purpose. The extensions to deal with knowledge acquisition also resulted in changes to the user interaction, particularly in the expert assigning further concepts (keywords) to documents and designing questions to assist the user in specifying the concepts they were interested in.

The system has two main functions. One is for expert(s) to build and maintain a knowledge base for the help documents. The expert can conduct their own search with a set of keywords to judge whether the retrieved documents have been incorrectly classified, keywords are missing from documents or a new document needs to be added. In adding a new document the expert distinguishes between the keywords of the new document and the retrieved documents satisfied by the keyword set that the expert used for the search. After adding the new document if a set of cornerstone cases exist, the expert should differentiate the cornerstone cases and the current case\textsuperscript{42} by adding new keyword(s).

The second function is for users to find the help documents constructed in the knowledge base. A user can search for information using one of the search methods supported by the system: “By Keyword”, “By Interaction” and “By Keyword and Interaction”. When the user is not satisfied with the retrieved documents, they can report their problem through the system. The reported problem is stored as a new unsolved case. An expert can then refine the knowledge base by diagnosis of the reported problems, and the knowledge base is gradually improved.

\textsuperscript{41} \url{http://www.cse.unsw.edu/faq/index.html} (2002).

\textsuperscript{42} The new document and its keywords comprise the current case.
3.2.2. Keywords and Help Questions

This help desk system uses a concept “keyword” to represent the meaning of the help documents. A keyword can be a representative word to express the meaning, purpose and role of a document in some way. The keyword may or may not exist in the content of a document. The key issue here is that human expert(s) decide the keywords for each help document, rather than automatically deriving them using some machine learning techniques. This should be a very large task if the expert had to decide on keywords for a whole body of documents.

With an RDR approach however, keywords are only added to a document when it has failed to be retrieved or retrieved inappropriately, and there are not already appropriate keywords to construct rules for the document to be retrieved correctly. As the expert adds keywords, s/he is also shown documents which might be retrieved by the same keywords, and the expert is asked to add keywords that distinguish the document that should be retrieved and the documents being inappropriately retrieved. Because of the contextual nature of the task, this is trivial for experts. Both RDR and Personal Construct Psychology (Gaines and Shaw 1990) are based on the fact that people find making distinctions between objects in a particular context very easy. But of course, some machine learning techniques can cooperate with the task of experts in assigning appropriate keywords for the help documents.

When documents are very similar, it is difficult to retrieve appropriate documents by automated information retrieval algorithms. However, human beings have little trouble in generating keywords that distinguish documents. The requirement, as for any expert system, is that experts have sufficient mastery of the domain to make reasonable distinctions between documents.

Keywords will be created with abstracted words so that to assist users the expert can attach a help question or an explanation sentence for each keyword. The idea of the help questions is to give an explanation (or definition) for each keyword in a form of human thinking way. For example, the keyword “change_login_shell” can be given a help question such as “How can I change my login shell?” to help users.
3.2.3. Knowledge Structure

The knowledge base of the system is stored in an MCRDR tree structure in which each node of the rule tree corresponds to a rule with a classification (i.e., a document). Figure 3.1 shows an example of the knowledge structure for the system. In an MCRDR tree, rules are allowed to have one or more conditions. For this study, we preferred to use rules with only a single condition\(^{43}\). Thus, if a document has more than one keyword, the second assigned keyword for the document becomes the child node of the first assigned keyword on the rule tree and so on. As a result of this, nodes which have no classification (i.e., no document) can exist in this structure. Note that the keywords of the help documents are used for the rule conditions of the knowledge base in the system.

In Figure 3.1, “a, b, c, d, e, f, g” are the keywords of the documents and are used for the conditions of the rule tree. Each node of the rule tree corresponds to a rule with a condition. A classification is essentially a link to an HTML document. The highlighted boxes represent rules that are satisfied for the test case with keywords {a, b, f}.

\[\text{Test Case: } \text{a, b, f}\]

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{knowledge_structure.png}
\caption{An example of the knowledge structure for the help system.}
\end{figure}

\(^{43}\) Note that there can exist a number of different strategies for storing rules in the MCRDR structure. The main reason for choosing this strategy (only use a single condition for a rule) was the option of using the knowledge structure for browsing. If a rule with a conjunction of conditions is built, rules would likely be added towards the top of knowledge structure (i.e., it would be a child rule of the root). Such a flat tree structure would not be suitable for browsing.
With MCRDR, all the rules in the first level of the rule tree for the given case (rule 1, 2 and 3 in Figure 3.1) are evaluated. Then, MCRDR evaluates the rules at the next level that are refinements of the rule satisfied at the top level and so on. Rules 1 and 2 are satisfied by the test case \{a, b, f\} so that the next rules to be evaluated are rules 4, 5 and 7 (i.e., refinements of rule 1). The process will stop when there are no more child nodes to be evaluated or none of these refinement rules are satisfied by the case. It can end up with more than one path for a particular case.

### 3.2.4. Knowledge Acquisition

An RDR knowledge base is built and maintained through the procedure of acquiring a correct classification, automatically deciding on a new rule’s location and acquiring rule conditions. The knowledge acquisition for this help system is achieved in the same way as the standard RDR knowledge acquisition mechanism. A new case is added into the knowledge base, when a user’s query is not satisfactorily handled (i.e., the case has been classified incorrectly or the case does not exist in the rule tree).

In this help desk system, when a user is not satisfied with the retrieved documents of their query, they can report their problem through the system. The reported problem is then passed on to a human expert as a new unsolved case with the log of information on the query used for the search and the documents retrieved by the query, and any free-text comments from the user outlining their problem. The expert can refine the knowledge base by an analysis of the reported problem and if necessary by e-mailing or talking to the user to find out what they really want. It is unlikely that all users will report their problems so that the system logs all users’ search activities. By analysing these search activities periodically, the expert can also refine the knowledge base.

In MCRDR knowledge acquisition, the system asks the expert to input or select conditions (keywords) and a conclusion (document) for the case\(^{44}\). After this procedure, the system searches all the stored cases which can satisfy the given conditions. If the system finds cases satisfying the conditions, the expert should distinguish between the

\(^{44}\) Again a case consists of a document and its keywords.
keywords of the new case and the keywords of the existing documents (cornerstone cases). The expert will be asked to select features (keywords) which can distinguish between the cornerstone cases and the new case. This process will be repeated until there is no cornerstone case which satisfies the new rule. Finally, the new case is stored in the rule tree as the refinement case of the previously wrongly retrieved document.

### 3.2.5. Search Methods

Users can retrieve the help documents using three different search methods: “By Keyword”, “By Interaction” and “By Keyword and Interaction (combined)”. The keyword method is based on the general information retrieval mechanisms. The interaction method is based on the MCRDR tree structure. That is, the inference process of MCRDR is utilised as a browsing mechanism. The last method is the combination of the keyword and the interaction method.

#### Keyword Method

With the keyword method, a user can select keywords provided by the system and/or can enter any textwords. The system provides a list of keywords that have been used as the keywords of the help documents. Both the rule conditions of the knowledge base as well as the contents of the documents are searched for the user’s query. The system uses simple keyword search techniques based on the basic Boolean operators (disjunction and conjunction).

For example, in Figure 3.2, when the user specifies “*printer*” as a search term, the documents corresponding to rules 6, 4, 8, and 9 will be retrieved. The document of rule 6 is retrieved as the content of the document contains the query “*printer*”. On the other hand, the documents of rules 4, 8 and 9 are retrieved as the condition of rule 4 is satisfied by the query. If there are any conditions satisfied by the search term, the conclusions (documents) of the rules and their refinement (child) rules are selected all together and presented to the user. This produces useful candidate documents although the search term is not included in the contexts of the help documents. If the user wants to get more specific documents among a search result, s/he can select further keywords with the conjunction operator.
**Interaction Method**

If the user is not good at identifying keywords or knows little about the domain, s/he may want to be guided by the system to get the information in a similar way to the directory or classification scheme of general search engines. The system here however, depends on interaction rather than an index structure. This means that the interaction is guided by the inference process of MCRDR. The user interacts with the system by selecting keywords listed by the system.

For example, when the user tries to find some documents using this method in Figure 3.2, the system will initially show the conditions (account, email, www, and printer) of the top-level rules (rules 1, 2, 3, and 4). The user can select some of these conditions to continue their search. Suppose that the conditions of rules 3 and 4 are selected. Then, the system will produce documents 3 and 4 as a search result and will show the conditions of rules 8 and 9 (refinement of rule 4) as possible further refinements. This process is repeated until the user finds the documents what they were looking for or there are no more child rules (refinement rules).

![Diagram](image)

**Figure 3.2.** The result documents by each search method with the keyword “printer”.

*The highlighted boxes (rules 6, 4, 8, and 9) show the rules resulting from the keyword search. The shadowed boxes (rules 1, 2, 3, and 4) are the first level rules shown with the interaction method. When a user tries to find the documents by the combined method with the keyword “printer”, the grey coloured boxes (rules 1 and 4) will be shown as the first level rules.*
**Combined Method**

The last method is the combination of the keyword and the interaction method. In the combined method, the system finds documents by the keyword method first (i.e., from both the contents of the help documents and the rule conditions of the KB). Then, the system organises the MCRDR rule tree with the conditions which contain the documents that satisfy the user query, and guide the user based on the conditions in the reorganised rule tree in the same way as the interaction method above.

In Figure 3.2, when the user tries to find documents with the keyword “printer” using the combined method, the conditions from the grey coloured boxes (rules 1 and 4) will be shown as the first level rules to be reviewed. Here, rules 2 and 3 are truncated compared to the interaction method so that with this combined method the system can reduce the number of interactions and conditions to be reviewed by the user compared to the interaction method alone.

### 3.2.6. Optimising Process of a Rule Tree

When a user finds documents using the combined method, the system optimises the rule tree to reduce the number of conditions to be reviewed by the user, and the number of interactions between the user and the system. Figure 3.3 shows an optimising process for a rule tree.

![Figure 3.3](image)

**Figure 3.3.** An optimising process of a rule tree.

(a): The original rule tree. (b): The optimised rule tree after deleting irrelevant paths from (a).

(c): A shortened rule tree from (b). (d): An alternative shortened rule tree from (b).
We suppose that Figure 3.3(a) is the original MCRDR rule tree. The shaded rules correspond to the keywords the user specifies. By ignoring the rule paths where no cases were selected, the subset of the MCRDR rule tree can be obtained as shown in Figure 3.3(b). Through this optimising process of the rule tree, the number of conditions that the user has to check is reduced.

The user can interact with this optimised rule tree (Figure 3.3(b)) to find documents. However, unnecessary interaction between the user and the system can exist in this rule tree, as nodes without the relevant keywords may still be included. For instance, documents 5, 8 and 2 are selected by the keyword search, but this does not imply that all the conditions of the rules in the path up to the point are satisfied. This means that some conditions may not be necessary to be checked. For example, the condition in rule 6 may not have the potential for the search interaction and this is simpler to just ask the condition in rule 8.

Consequently, to reduce the number of interactions the system regenerates the optimised rule tree to the shortened form as shown in Figure 3.3(c). There can exist a number of different ways to shorten the optimised rule tree. Figure 3.3(d) can be an alternative shortened tree. A difference between Figure 3.3(c) and Figure 3.3(d) is the number of interactions to be reviewed by the user. With Figure 3.3(d) one more interaction time can be reduced compared to Figure 3.3(c). On the other hand, the number of conditions to be checked is increased at the top level. If we put all the selected rules as the child rules of the root rule as shown in Figure 3.3(d) (except the parent and its children node are selected altogether. In this case the original rule tree structure is kept), it can cause the problem of an increase in the number of conditions to be checked again. The greater the size of knowledge base, the more serious this problem becomes.

In order to improve this problem we used the following strategy: The rules which are not selected by the specified keywords are all truncated except for child rules of the root node that have fired. Based on this strategy, rule 1 is selected and rule 6 is truncated in Figure 3.3(b). When the parent and its child node are selected all together, the original rule tree structure is kept. For example, in Figure 3.3(b), if rules 6 (parent node of rule
8) and 8 (child node of rule 6) are selected, the relationship structure (parent and child) is kept. Through this process the number of interactions and the number of conditions checked by users can be reduced. However, options to explore other branches of the rule tree can be lost in this optimised and shortened tree. That is, there can exist a trade-off between the number of interactions reduced and possible relevant branches lost in this approach. When a user formulates a query with inappropriate keywords, this trade-off will be entailed. However, options to explore other branches of the rule tree can be kept using a proper browsing interface (i.e., using different colours for folders).

The implementation and interfaces of the system can be found in (Kim et al. 1999).

3.3. Conclusion and Discussion

This study has taken a step in the direction of finding a new approach to information retrieval maintenance based on incrementally developing a knowledge base. The earlier work of Kang et al. (1997) utilised an RDR structure for information retrieval and developed search methods based on the RDR structure. These suggestions have been refined in this study focusing on an incremental knowledge acquisition process and a prototype system for the FAQ page of a School of Computer Science. The central insight in this study is that an RDR system can be used as a mechanism for indexing and retrieving documents, whether these are actual documents or expert opinions especially constructed for the KBS. The second major insight that has motivated this application, is that the same problems of situated cognition that apply to conventional KBS also apply to building document retrieval systems for particular domains.

This RDR help desk system allows users to enter words that might occur in a document. It then retrieves documents relevant to the words entered. If only one document is retrieved the search is over. If not, an interactive session with the user is commenced. The documents retrieved are all “conclusions” of various rules in the system so that a subset of the rules needed to refine the search, can be identified via the documents retrieved. The system then asks the user to provide further information to identify which rule applies. For example, if the user is seeking information on a printer queue, they might be asked whether they want to delete a document from a queue, or estimate when
their material might be printed. The further information requested identifies which rule conditions are satisfied and which rules can fire. Of course further “textwords” can be added during the interaction to narrow the search.

If the interaction fails to deliver an appropriate document, the query is referred to a human expert - the actual human operator of a manned help desk or their supervisor. The user can transmit a free text comment to the expert indicating the nature of their query and if necessary there can be an interaction between the user and the expert to clarify the nature of the query. This is not an onerous mechanism and is used in all sorts of circumstance. In addition, the expert is provided with a trace of the previous interaction with the automated help desk. This should be more than enough information to allow the expert to add rules to ensure the correct document is retrieved next time. The rules simply ask the user whether they are interested in a particular concept and go through a series of concepts until the document is retrieved.

However, there are a number of questions still to be answered. Firstly, the system with the RDR techniques has not been yet evaluated in routine use, even though there is some commercial use in help desk support. Thus, a strong conclusion about the ergonomic suitability of the method cannot be made. Similarly, knowledge acquisition has not yet been carried out based on logged and referred queries. There are again significant ergonomic issues, on whether the experts find the information provided is adequate for further rules and are happy to add new rules in a timely manner so users will not become disillusioned. From the experience with RDR elsewhere, we have every expectation that rules will be easy to add, but this needs to be evaluated. What we developed was a prototype demonstration that this type of information retrieval and information retrieval maintenance is possible.

Secondly, the initial mode of interacting with the system is for a user to enter a conjunction of terms. If this does not produce the correct documents it is followed by an interactive session where the system leads the user through a dialogue to refine their search. It is anticipated that it would also be helpful to have some natural language

---

45 Byeong Kang, personal communication.
processing (NLP), particularly for the initial entry. We believe that again RDR may usefully be applied to this as we are dealing with narrow domains and therefore a reasonably compact language, even when the user is naive and does not know many of the domain terms. Research in this area will need to focus on refining the natural language text form of the input query into a more standard set of features to match the cases in the RDR case base. Natural language forms have been used in diagnostic systems (Anick 1993; Barletta 1993b; Burke et al. 1997; Katz 1997).

Thirdly, it is likely that we may further refine the MCRDR structure for this type of domain (information retrieval). Initially RDR were developed for domains with attribute-value data with perhaps hundreds of attributes with numerical data and a smaller number of attributes with a small number of enumerated values. Here the keywords represent a potentially huge number of Boolean attributes. However, one of the key assumptions underlying this work was to develop a help desk system for fairly small and specialised domains. The goal of the system was to make it easy to develop a specialist information retrieval system for these domains, as a general search engine could not devote the effort to getting a right term for each domain.

There is an important contrast between the HDS and previous RDR applications which needs to be addressed. With previous RDR systems, all the relevant data was known at the start of the inference and what mattered was the conclusion rather than the inference path by which it was reached. Here, however, the structure of RDR is used for browsing (information seeking tasks) to guide the user interaction. That is, the main difference between the HDS and previous RDR applications is that the user is supposed to see and browse the knowledge structure of RDR. The RDR approach was initially developed for knowledge acquisition for knowledge based systems (Compton and Jansen 1990). It has been applied to a range of tasks, but is best known for its use in providing clinical interpretations for Chemical Pathology reports (Edwards et al. 1993). In Chemical Pathology all the data is provided by a laboratory information system, so that reports can be generated without user involvement; while the task of the expert in adding rules is simply to identify significant features in the data.
On the other hand, information retrieval requires user interaction and there are problems with this. The user can either enter keywords or respond to queries about keywords. Users often prefer some sort of browsing mechanism rather than responding to queries. As well, the ordering of the keywords presented in RDR reflects the historical development of the system, not the most natural order for the user. Although, as demonstrated in other RDR work, RDR greatly assists context-specific knowledge acquisition, it does not organise the knowledge in a way that is suitable for browsing. One might consider that an evolving RDR system produces a type of hierarchy and that this will be adequate, as experts tend to provide general rules first (Suryanto et al. 1999). However, this does not necessarily mean that documents that are more general or more introductory will be found higher up the tree, or that neighbouring documents in the tree are necessarily appropriate neighbours.

3.4. Chapter Summary

By developing the help desk system, the possibility of a new way of information retrieval was demonstrated where an expert can rapidly build and maintain an information retrieval, or a help desk system in his or her area of expertise based on the RDR techniques. The methodology of RDR for information retrieval is somewhat different from the use of RDR in other areas. Here, the structure of RDR is used to guide the user interaction. However, even though RDR greatly assists incremental and context-specific knowledge acquisition, and a robust maintenance process, it does not organise the knowledge in a way that is suitable for browsing.

Therefore, more studies need to be conducted to explore some mechanisms for reorganising the RDR tree to make it appropriate for browsing. Hierarchical structures for organising relations between terms in a domain and associated information, or structures for organising documents, are increasingly referred to as ontologies. Thus, in the longer term we believe that such a hierarchy will need to be reorganised and that many different structures will be possible depending on different ontological frameworks. A proper browsing scheme will be required to access these different ontologies.
The next chapter presents the basic notions of Formal Concept Analysis, which is the core technology in the proposed system. The key strategy of the proposed system is to incorporate the advantages of the concept lattice of Formal Concept Analysis (FCA) appropriate for browsing, while keeping the incremental aspects of Ripple-Down Rules (RDR). FCA has previously been used with RDR expert systems as an explanation tool (Richards and Compton 1997a).
Chapter 4
Formal Concept Analysis

Formal Concept Analysis (FCA) was developed by Rudolf Wille in 1982 (Wille 1982). It is a theory of data analysis which identifies conceptual structures among data sets based on the philosophical understanding of a “concept” as a unit of thought comprising its extension and intension as a way of modelling a domain (Wille 1982; Ganter and Wille 1999). The extension of a concept is formed by all objects to which the concept applies and the intension consists of all attributes existing in those objects. These generate a conceptual hierarchy for the domain by finding all possible formal concepts which reflect a certain relationship between attributes and objects. The resulting subconcept-superconcept relationships between formal concepts are expressed in a concept lattice which can be seen as a semantic net providing “hierarchical conceptual clustering of the objects… and a representation of all implications between the attributes” (Wille 1992, pp.493). The implicit and explicit representation of the data allows a meaningful and comprehensible interpretation of the information.

The method of FCA has been successfully applied to a wide range of applications in medicine (Cole and Eklund 1996b), psychology (Spangenberg et al. 1999), ecology (Brüggemann et al. 1997), civil engineering (Kollew et al. 1994), software engineering (Lindig and Snelting 2000; Snelting 2000), library (Rock and Wille 2000) and information science (Eklund et al. 2000). A variety of methods for data analysis and knowledge discovery in databases have also been proposed based on the techniques of FCA (Stumme et al. 1998; Hereth et al. 2000; Wille 2001). Information Retrieval is also a typical application area of FCA (Godin et al. 1993; Carpineto and Romano 1996a; Priss 2000b; Cole and Stumme 2000; Cole and Eklund 2001).

This chapter is organised as follows: Section 4.1 introduces the basic notions of Formal Concept Analysis. Section 4.2 describes the concept lattice of FCA, and surveys a
number of algorithms in the literature for constructing a concept lattice from a context. Conceptual scaling, a technique for dealing with many-valued contexts is explained in Section 4.3. Finally, Section 4.4 reviews lattice-based information retrieval, as the aim of this thesis is to develop domain-specific document retrieval mechanisms based on the FCA techniques.

4.1. Basic Notions of FCA

This section describes the basic notions of FCA such as formal contexts and formal concepts. The formulas used in this chapter closely adhere to the notions of Wille (1982) and Ganter and Wille (1999). The adjective “formal” emphasises that Formal Concept Analysis methods deal with mathematical notions (Ganter and Wille 1999, pp17). Here, the word context and concept are used to denote a formal context and formal concept, respectively.

4.1.1. Formal Context

The most basic data structure of FCA is a formal context $K := (G, M, I)$ which consists of two sets $G$ and $M$, and a binary relation $I$ between $G$ and $M$. The elements of $G$ and $M$ are called the objects and attributes of the context, respectively. The relation $I$ indicates where an object $g$ has an attribute $m$ by the relationship $(g, m) \in I$. The relation $I$ sometimes is written $gIm$.

Table 4.1 shows an example of the formal context $(G, M, I)$ for a part of “the Animal Kingdom”. Here, the objects are animals and the attributes are the properties of the objects. A context is normally represented by a cross table with the object names in the rows and the attribute names in the columns.

Table 4.1 shows an example of the formal context $(G, M, I)$ for a part of “the Animal Kingdom”. Here, the objects are animals and the attributes are the properties of the objects. A context is normally represented by a cross table with the object names in the rows and the attribute names in the columns.

In Table 4.1, the context $(G, M, I)$ consists of a set of objects $G$ \{cheetah, tiger, giraffe, ostrich, penguin\} and a set of attributes $M$ \{has hair, has feathers, eats meat, has dark spots, can swim, has long neck\} where the relation $I$ is \{(cheetah, has hair), (cheetah, eats meat), (cheetah, has dark spots), ..., (ostrich, has feathers), (ostrich, has long neck), (penguin, has feathers), (penguin, can swim)\}.
Table 4.1. Formal context for a part of “the Animal Kingdom”.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cheetah</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>Tiger</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Giraffe</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>Ostrich</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>Penguin</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

A symbol “X” designates that a particular object has the corresponding attributes.

4.1.2. Formal Concept

Formal concepts reflect a relationship between objects and attributes. A formal concept is defined as a pair $(X, Y)$ where $X$ is the set of objects and $Y$ is the set of attributes. The set $X$ is called the extent and the set $Y$ is called the intent of the concept $(X, Y)$. The following derivation operators are used to compute formal concepts of a context.

For any set $X \subseteq G$ and any set $Y \subseteq M$, $X'$ and $Y'$ are defined correspondingly as follows: $X' := \{ m \in M \mid \forall g \in X: (g, m) \in I \}$ and $Y' := \{ g \in G \mid \forall m \in Y: (g, m) \in I \}$. Then, a formal concept is formulated as a pair $(X, Y)$ with $X \subseteq G, Y \subseteq M, X' = Y$ and $Y' = X$.

The formulas 4.1 and 4.2 can be used to construct all formal concepts of a context denoted by $\mathcal{B}(G, M, I)$. First, all row-intents $\{g\}'$ with $g \in G$ (formula 4.1) or all column-extents $\{m\}'$ with $m \in M$ (formula 4.2) are obtained. Then, all their intersections are found so that all extents $X'$ or all intents $Y'$ of the formal concepts of $K$ can be determined. Following this, the intents of all determined extents are computed. Note that there are a number of different algorithms in the literature.

$$X' = \bigcap_{g \in X} \{g\}' \quad (4.1) \quad Y' = \bigcap_{m \in Y} \{m\}' \quad (4.2)$$

Table 4.2 shows an example of how all the concepts can be drawn from the context $(G, M, I)$ in Table 4.1 based on formula 4.2. The detailed process is as follows. Note that this process is based on the formulae of Wille (1982).
### Table 4.2. A procedure of finding formal concepts from the context in Table 4.1.

<table>
<thead>
<tr>
<th>Step</th>
<th>Intent</th>
<th>Extent</th>
<th>Step</th>
<th>Intent</th>
<th>Extent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>{1, 2, 3, 4, 5}</td>
<td>1</td>
<td></td>
<td>{1, 2, 3, 4, 5}</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>{1, 2, 3}</td>
<td>2</td>
<td>a</td>
<td>{1, 2, 3}</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>{4, 5}</td>
<td>3</td>
<td>b</td>
<td>{4, 5}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{}</td>
<td></td>
<td></td>
<td>{}</td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>{1, 2}</td>
<td>4</td>
<td>c</td>
<td>{1, 2}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>{1, 3}</td>
<td>5</td>
<td>d</td>
<td>{1, 3}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{1}</td>
<td></td>
<td></td>
<td>{1}</td>
</tr>
<tr>
<td>6</td>
<td>e</td>
<td>{5}</td>
<td>6</td>
<td>e</td>
<td>{5}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>f</td>
<td>{3, 4}</td>
<td>7</td>
<td>f</td>
<td>{3, 4}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{3}</td>
<td></td>
<td></td>
<td>{3}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{4}</td>
<td></td>
<td></td>
<td>{4}</td>
</tr>
</tbody>
</table>

(a) (b)

**Procedure 1**: Formulate an extent containing the set of objects $G$ representing the largest concept of $K$. Then, perform Procedure 2 for each attribute $m$.  

**Procedure 2**: Find the set of objects $X$ which contains the attribute $m$. Following that, check whether any extent in the list is equivalent to $X$. If an equivalent extent of $X$ does not exist in the list, the set $X$ is added as an extent of the attribute. Next, the intersection of $X$ and all extents calculated in previous steps, is determined. When the intersection set does not exist in the list, the set is also added as an extent of the attribute. Table 4.2(a) shows the result of Procedure 1 and 2.

**Procedure 3**: Then, for each extent in Table 4.2 (a), its intent $Y \leftarrow \{m \in M \mid glm \text{ for all } g \in X\}$ is determined. Table 4.2 (b) shows the result of this step. Now we have 11 formal concepts for the context $(G, M, I)$ in Table 4.1.

### 4.2. Concept Lattice

The formal concepts of a context $K$ are expressed in a concept lattice which provides hierarchical conceptual clustering of the objects and a representation of all implications between the attributes (Wille 1992).
4.2.1. Construction of a Concept Lattice

The concept lattice is the basic conceptual structure of FCA ordered by the smallest set of attributes (intent) between the concepts. To form a concept lattice, hierarchical subconcept-superconcept relations between all the formal concepts need to be found. This is formalised by $(X_1, Y_1) \leq (X_2, Y_2) : \iff X_1 \subseteq X_2 (\iff Y_2 \subseteq Y_1)$ where $(X_1, Y_1)$ is called a subconcept of $(X_2, Y_2)$ and $(X_2, Y_2)$ is called a superconcept of $(X_1, Y_1)$. The relation $\leq$ is called the hierarchical order of the concepts. The set of all the formal concepts of the context $(G, M, I)$ with this ordered relation is a complete lattice in which the infimum and the supremum are given by formula 4.3 and 4.4.

$$\bigvee_{i \in I} (X_i, Y_i) := \big((\bigcup_{i \in I} X_i)^*, \bigcap_{i \in I} Y_i\big) \quad (4.3)$$
$$\bigwedge_{i \in I} (X_i, Y_i) := \big((\bigcap_{i \in I} X_i, (\bigcup_{i \in I} Y_i)^*) \quad (4.4)$$

The complete hierarchical subconcept-superconcept relation is called the concept lattice of the context $(G, M, I)$ denoted by $\mathcal{E}(G, M, I)$. The line diagram in Figure 4.1 shows the concept lattice of the context $K$ in Table 4.1. Each node represents a formal concept $(X, Y)$. Not only all relations between objects and attributes but also all relations between objects and between attributes can easily be observed through this lattice.
4.2.2. Algorithms for Constructing a Concept Lattice

Computing a concept lattice is an important issue and has been widely studied to develop more efficient algorithms. As a consequence, a number of batch algorithms (Chein 1969; Ganter 1984; Bordat 1986; Ganter and Reuter 1991; Kuznetsov 1993; Lindig 1999) and incremental algorithms (Norris 1978; Dowling 1993; Godin et al. 1995; Carpineto and Romano 1996a; Ganter and Kuznetsov 1998; Nourine and Raynaud 1999; Stumme et al. 2000; Valtchev and Missaoui 2001) exist in the literature. Batch algorithms build formal concepts and a concept lattice from the whole context in a bottom-up approach (from the maximal extent or intent to the minimal one) or a top-down approach (from the minimal extent or intent to the maximal one). Incremental algorithms gradually reformulate the concept lattice starting from a single object with its attribute set.

Godin et al. (1995) demonstrated that even the simplest and least efficient, incremental algorithms outperformed all the batch algorithms in their experimental comparative study. Recently, Kuznetsov and Ob’edkov (2001)\textsuperscript{46} conducted another comparative study, both theoretical and experimental. The study found that the performance of algorithms depends on the properties of input data such as the size of contexts and the density of contexts. Results indicated that the Godin algorithm (Godin et al. 1995) was a good choice in the case of small and sparse contexts. On the other hand, the Bordat algorithm (1986) showed a good performance for large, average density contexts. However, when the set of objects was small, the Bordat algorithm was several times slower than other algorithms. The study also indicated that the Kuznetsov algorithm (1993) and the Norris algorithm (1978) should be used for large and dense contexts.

More recently, the Titanic algorithm (Stumme et al. 2000) and the Valtchev algorithm (Valtchev and Missaoui 2001) have been released and have not been included in the above comparative study. The Titanic algorithm is based on data mining techniques for computing frequent item sets. The experimental results showed that the algorithm is

faster than Ganter’s Next-Closure (Ganter and Reuter 1991) for the whole data set under normal conditions.

The Valtchev algorithm extended the Godin et al. algorithm (1995) based on two scenarios. In the first scenario, the algorithm updates the initial lattice by considering new objects one at a time. In the second one, it builds the partial lattice over the new object set first and then merges it with the initial lattice. The first algorithm showed an improvement of the Godin et al. algorithm (1995) and was suggested for small sets of objects. On the other hand, the second was suggested as the right choice for medium-size sets.

Table 4.3 shows a summary of the time complexity and polynomial delay of algorithms.\(^{47}\) \(|G|\) denotes the number of objects, \(|M|\) the number of attributes and \(|L|\) the size of the concept lattice. In the Godin algorithm (1995), \(\mu\) designates an upper bound on \(|f(\{x\})|\) where the set of objects associated with the attribute \(x\) is denoted by \(f(\{x\})\). When there is a fixed upper bound \(\mu\), the time complexity of this algorithm is \(O(|G|)\). The Nourine algorithm (1999) is half-incremental. It incrementally constructs the concept set, but formulates the lattice graph in batch.

| Table 4.3. Summary for the time complexity and polynomial delay of algorithms. |
|---------------------------------|-------------------|-------------------|-------------------|
| Incremental | Time complexity | Polynomial delay |
| Ganter (1984) | \(O(|G|^2|M||L|)\) | \(O(|G|^3|M|)\) |
| Bordat (1986) | \(O(|G||M|^2||L|)\) | \(O(|G||M|^3|)\) |
| Kuznetsov (1993) | \(O(|G|^2|M||L|)\) | \(O(|G|^3|M|)\) |
| Lindig (1999) | \(O(|G|^2|M||L|)\) | \(O(|G|^3|M|)\) |
| Norris (1978) | X | \(O(|G|^2|M||L|)\) | N/A |
| Dowling (1993) | X | \(O(|G|^2|M||L|)\) | N/A |
| Godin (1995) | X | \(O(2^\mu|G|)\) | N/A |
| Nourine (1999) | X | \(O(|G| + |M||G||L|)\) | N/A |
| Titanic (2000) | X | \(O(|G|^2|M||L|)\) | N/A |
| Valtchev (2001) | X | \(O(|G| + |M||G||L|)\) | N/A |

\(^{47}\) Note that not all of the algorithms are indicated in the table.
The main issues for computing all the formal concepts of a context are related to how all the concepts of a context without the repetitive generation of the same concept can be generated. There are a number of techniques to avoid repetitive generation of the same concept - divide the set of concepts into several parts, use a hash function, maintain an auxiliary tree structure or use an attribute cache.

Kuznetsov and Ob’edkov (2001) noted that an empirical comparison of algorithms is not an easy task for a number of reasons. First of all, algorithms described by authors are often unclear, leading to misinterpretations. Secondly, the data structures of the algorithms and their realisations are often not specified. Another issue is related to the setting up of consensus data sets to use as test beds. The context parameters such as an average attribute set associated with an object and vice versa, or the size and density of contexts should be considered. The test environments such as programming languages, implementation techniques, and platforms are also crucial factors which influence the performance of algorithms. For example, Valtchev and Missaoui (2001) indicated that the Nourine algorithm (1999) is the most efficient batch algorithm, while Kuznetsov and Ob’edkov (2001) indicated that the Nourine algorithm is not the fastest algorithm, even in the worst case. Kuznetsov and Ob’edkov (2001) noted that this result was probably caused by different implementation techniques. These are the main reasons for the existence of quite a lot of algorithms in the literature.

### 4.3. Conceptual Scaling

Conceptual scaling has been initiated in order to deal with many-valued attributes (Ganter and Wille 1989; Ganter and Wille 1999). Usually more than one attribute consists of an application domain and each attribute with a range of values so that there is a need to handle many attributes in a context. In addition, often there is a need to analyse (or interpret) concepts in regard to interrelationships between attributes in a domain. This is the main motivation for conceptual scaling.

---


49 E. Mephu Nguifo et al. (Eds.): CLKDD’01, pp. 43.
For instance, the domain of a “used car market” consists of a number of attributes such as price, year built, maker, colour, body type, transmission and others, and each attribute with a set of values. Such attributes can present all together in a context named with a many-valued context. Then, when one is interested in analysing “used cars” regarding an interrelationship between certain attributes in the many-valued context, they can combine the attributes of interest into a concept lattice.

A many-valued context is defined as $K = (G, M, W, I)$ which consists of sets $G, M, W$ and a ternary relation $I$ between $G, M$ and $W (I \subseteq G \times M \times W)$. The elements of $G,$ and $M$ are called the objects and attributes of $K$ respectively, and the elements of $W$ attribute values. The notion $(g, m, w) \in I$ indicates the attribute $m$ has the value $w$ for the object $g$.

A many-valued context can be represented in a table which is labelled by the objects in the rows and by the attributes in the columns. Table 4.4 shows an example of a many-valued context for the domain of a “used car market”. The context $(G, M, W, I)$ consists of a set of objects $G$ \{car1, car2, car3, car4, car5, car6\}, a set of attributes $M$ \{maker, transmission, body type, colour, price\} and a set of attribute values $W$ \{DAEWOO, HYUNDAI, KIA, SSANGYOUNG, Auto, Manual, Hatch back, … , $5,000, $7,000, $9,000, $11,000, $14,000, $16,000\}. An entry in row $g$ and in column $m$ designates the attribute value $w$.

<table>
<thead>
<tr>
<th>Car</th>
<th>Maker</th>
<th>Transmission</th>
<th>Body type</th>
<th>Colour</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car 1</td>
<td>DAEWOO</td>
<td>Auto</td>
<td>Hatch back</td>
<td>White</td>
<td>$5,000</td>
</tr>
<tr>
<td>Car 2</td>
<td>HYUNDAI</td>
<td>Manual</td>
<td>Sedan</td>
<td>Silver</td>
<td>$7,000</td>
</tr>
<tr>
<td>Car 3</td>
<td>KIA</td>
<td>Auto</td>
<td>Convertible</td>
<td>Burgundy</td>
<td>$9,000</td>
</tr>
<tr>
<td>Car 4</td>
<td>DAEWOO</td>
<td>Manual</td>
<td>Sedan</td>
<td>Red</td>
<td>$11,000</td>
</tr>
<tr>
<td>Car 5</td>
<td>HYUNDAI</td>
<td>Auto</td>
<td>Coupe</td>
<td>Black</td>
<td>$14,000</td>
</tr>
<tr>
<td>Car 6</td>
<td>SSANGYOUNG</td>
<td>Auto</td>
<td>Wagon</td>
<td>Silver</td>
<td>$16,000</td>
</tr>
</tbody>
</table>

**Table 4.4.** An example of a many-valued context for a part of a “used car market”. 
Each attribute of the many-valued context can be transformed into a one-valued context called a conceptual scale. Then, the scales are joined together as a way of interpreting the concepts of objects. This interpretation process is called conceptual scaling.

A conceptual scale for a particular attribute \( m \in M \) of a many-valued context is defined as a context \( S_m := (G_m, M_m, I_m) \) where \( M_m \subseteq W \) is a set of values of the attribute \( m \) in the many-valued context \( K = (G, M, W, I) \) and \( G_m \subseteq M_m \).

Figure 4.2 and 4.3 show an example of a scale for the attribute \textit{price} and \textit{transmission} in Table 4.4. A scale context is equivalent to a one-valued context. In a scale context, both the rows and columns of the table are usually headed by the values of the scale attribute (e.g., Figure 4.3). However, any expression or interpretation of the values of attributes can be used to make it easier to define a scale especially for numerical attributes (e.g., Figure 4.2). The expressions can denote a range of values for the values of the scale attribute. To represent these expressions, Cole and Eklund (2001) introduce a function called the composition operator: For an attribute \( m \), \( \alpha_m : W_m \rightarrow G_m \) where \( W_m = \{ w \in W \mid \exists g \in G : (g, m, w) \in I \} \). This maps the values of the attribute \( m \) to scale objects.

<table>
<thead>
<tr>
<th>Price</th>
<th>Cheap</th>
<th>Mid-range</th>
<th>Expensive</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;$5,000</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$5,000 - $8,000</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>$8,000 - $12,000</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$12,000 - $15,000</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>&gt;$15,000</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

\textbf{Figure 4.2.} A scale context for the attribute \textit{price} (\( S_{\text{price}} \)) in Table 4.4 and its concept lattice.

\textit{Note that the scale context for the attribute “price” uses expressions rather than attribute values (i.e., \( \leq 8,000 = \text{cheap} \), \( \geq 8,000 \) \& \( \leq 15,000 = \text{mid-range} \), \( \geq 15,000 = \text{expensive} \)). A symbol “X” designates that the row value corresponds to the column value.}

<table>
<thead>
<tr>
<th>Transmission</th>
<th>Auto</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Manual</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

\textbf{Figure 4.3.} A scale context for the attribute \textit{transmission} (\( S_{\text{trans}} \)) and its concept lattice.
Table 4.5. A realised scale context for the scale price in Figure 4.2.

<table>
<thead>
<tr>
<th>Object</th>
<th>Cheap</th>
<th>Mid-range</th>
<th>Expensive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Car2</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Car3</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Car4</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Car5</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Car6</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Then, a realised scale can be driven from scales and the many-valued context when a diagram is requested at run time. Table 4.5 shows an example of this realised scale context for the attribute price. This is formulated from the scale for the attribute price in Figure 4.2, and its objects and values in the many-valued context in Table 4.4. In essence, the derived context is equivalent to a formal context presented in Section 4.1.1.

Figure 4.4 shows the derived concept lattice for the realised scale context in Table 4.5. A realised scale can be combined into this concept lattice to analyse concepts according to an interrelationship between two scales.

Figure 4.5 shows a combination of two scales in a lattice structure using a nested line diagram. The outer structure is the scale of price and the nested inner structure is the scale of transmission. Concepts of the many-valued context can be interpreted in this combined concept lattice. For instance, it can be read that there is no “manual” car in the “expensive” concept and no “auto” car in the “cheap | mid-range” concept. Note that the small grey vertex indicates that there is no object which satisfies the attribute value in the vertex, as opposed to the black vertex.
More than one attribute in a many-valued context can be combined in a scale. Conceptual scaling is also used with one-valued contexts in order to reduce the complexity of the visualisation (Stumme 1999; Cole and Stumme 2000). In this case, scales are applied for grouped vertical slices of a large context. More cases for the use of conceptual scaling can be referred to (Stumme 1999; Cole and Stumme 2000; Cole and Eklund 2001) and TOSCANA\(^{50}\) (Vogt et al. 1991; Kollewe et al. 1994; Vogt and Wille 1995).

4.4. FCA for Information Retrieval

Formal Concept Analysis has numerous applications for data analysis and information retrieval in fields such as medicine (Cole and Eklund 1996b), psychology (Spangenberg and Wolff 1991; Strahringer and Wille 1993; Spangenberg et al. 1999), ecology (Brügge, Schwaiger et al. 1995; Brügge, Zelles et al. 1995; Brügge et al. 1997), social science (Ganter and Wille 1989), and political science (Vogt et al. 1991). There are also applications of FCA in civil engineering (Kollewe et al. 1994), software engineering (Lindig 1995; Snelting 1996; Lindig and Snelting 2000; Snelting

---

\(^{50}\) TOSCANA is a software tool set for the visualisation of data with nested line diagrams and for navigating and the retrieval of objects in databases.

2000), linguistics (Grosskopf and Harras 1998), libraries (Rock and Wille 2000), and information science (Eklund et al. 2000). Most of these application systems elaborate the standard software tool, TOSCANA, which has been developed for analysing and exploring data based on the methods of FCA. There are a number of other lines of FCA research for knowledge representation with Conceptual Graphs (Wille 1997; Mineau et al. 1999), text data mining (Groh and Eklund 1999) and knowledge discovery in databases (Stumme et al. 1998; Hereth et al. 2000; Wille 2001).

Information Retrieval is one typical application area of FCA. A strong feature that makes FCA applicable to the field of information retrieval is that FCA can produce a visible concept lattice, which shows the inherent structure among data in a lattice so that it can be used as a classification system. A concept lattice of FCA represents the generalisation and specialisation relationships between document sets and attribute sets. Thus, the lattice of FCA can represent conceptual hierarchies for the applied domain. Moreover, it can be superior to the hierarchical tree structure as the lattice gives all minimal refinements and minimal enlargements for a query (Godin et al. 1995). In addition, the hierarchical tree structure, in which each cluster has exactly one parent, can be also embedded into the lattice structure.

With these advantages, a number of researchers have proposed the lattice structure for document retrieval (Godin et al. 1993; Carpineto and Romano 1996a; 1996b). More recently several researchers have also studied lattice-based information retrieval with graphically represented lattices along with nested line diagrams (Cole and Stumme 2000; Priss 2000b; Cole and Eklund 2001).

4.4.1. Godin et al. Approach

Godin et al. (1993) studied the advantage of the lattice method against hierarchical classification, and also evaluated retrieval performance by comparing the lattice structure with a manually built hierarchical classification and a conventional Boolean retrieval method. The performance of hierarchical classification retrieval showed significantly lower recall compared to the lattice-based retrieval and Boolean querying. No significant performance difference was found between the lattice-based retrieval and
Boolean querying, but the lattice structure was suggested as being an attractive alternative because of the potential advantage of lattice browsing. The experiments were performed on a small database extracted from a catalogue of films assigning a set of controlled terms manually to each film in the database. The prototype interface was implemented on a standard screen for a Macintosh microcomputer using window, menu and dialog interface tools and viewing only direct neighbours in the lattice.

4.4.2. Carpineto and Romano Approach

Carpineto and Romano (1995; 1996b) determined that the performance of lattice retrieval was comparable to or better than Boolean retrieval on a medium-sized database for a computer engineering collection which was assigned controlled terms manually. They (Carpineto and Romano 1996a) also extended their study using a thesaurus as background knowledge in formulating a browsing structure of FCA and presented experimental evidence of a substantial improvement after the introduction of the thesaurus. The interface developed by Carpineto and Romano (1995; 1996b) showed the lattice graph using a similar fisheye view technique (Furnas 1986) of individual nodes on a stand-alone Symbolic Lisp Machine. A Boolean query interface (Carpineto and Romano 1998) was also supported to move directly to a relevant portion of the lattice from a user’s query. It allowed users to navigate the lattice dynamically and made it easy to refine the query.

4.4.3. FaIR Approach

More recently, FCA has been used for document retrieval culminating in a faceted information retrieval system (FaIR) that incorporates a lattice-based faceted thesaurus (Priss 2000a; 2000b). In this approach, a thesaurus is predefined for an applied domain and divided into a number of facets\(^{52}\), called a faceted thesaurus. A portion of a hierarchy in the thesaurus can be a facet and it is represented with a lattice rather than a

---

51 A fisheye view (Furnas 1986) is a technique to view a specific portion of information in great detail while also showing the context that contains the detail.

52 “Facets are relational structure consisting of units, relations and other facets selected for a certain purpose” (Priss 2000a).
hierarchical tree structure. Then, documents are indexed into the concepts of the facet lattices by mapping the keywords of the documents to the facet concepts (i.e., by mapping functions). A document can be indexed into more than one facet lattice. Documents are retrieved by selecting a facet and a concept of the facet. A main facet lattice is then provided with the retrieved documents. Other facet lattices relevant to the retrieved documents are also displayed along with the main facet lattice if exist. The main advantage of FaIR is that large retrieval sets can be divided into smaller sets (i.e., facets) in a retrieval display. In addition, a set of concepts can be retrieved in response to a query from conceptual relationships among terms that are inherent to the domain thesaurus. The system (FaIR) described in the paper (Priss 2000b) is under development and a navigating interface is not yet published.

4.4.4. Cole et al. Approach

The focus in Godin et al. (1993), and Carpineto and Romano (1995; 1996b) was to examine the advantages and capabilities of lattice-based retrieval against a conventional Boolean querying and hierarchical classification retrieval. Cole et al. (Cole and Stumme 2000; Cole et al. 2000; Cole and Eklund 2001) have further developed more precise browsing mechanisms by combining conceptual scales using nested line diagrams for an e-mail management system (CEM) and a real estate system.

CEM (Cole and Stumme 2000; Cole et al. 2000) uses the concept lattice of FCA to organise and browse e-mails rather than a typical tree structure. It is based on TOSCANa, but a user can maintain and update an e-mail collection instead of a knowledge engineer. Each e-mail is assigned a set of catchwords. A hierarchy, a partially ordered set, is comprised of more general catchwords. Even though the hierarchy is represented by a tree, the embedded structure is a concept lattice. E-mails are managed based on the hierarchy. A cluster in the hierarchy can be a scale (i.e., default scale) and other scales can also be formulated to group related catchwords together. In assigning catchwords to an e-mail, the system identifies relevant catchwords to the e-mail from the general catchwords used for the hierarchy. The catchwords in the clusters of the hierarchy which include the identified relevant catchwords are added automatically as the catchwords for the e-mail. Other specific
catchwords are also added to the e-mail. In response to a user’s query, a virtual folder represented in a lattice structure is formulated with a collection of e-mail documents retrieved. Then, the user can navigate e-mails in the conceptual space by a scale in a simple line diagram or by combining scales in nested line diagrams.

CEM has been extended into a system for real-estate advertisements (Cole and Eklund 2001). CEM used conceptual scaling to deal with a one-valued context (i.e., in one attribute catchword), whereas the real estate system used it for a many-valued context (i.e., many attributes such as number of bedrooms, rental price, views and others). In this system, attributes and their values for the advertisements are pre-defined, and are presented in an ordered hierarchy. Scales are also predefined for each attribute in the hierarchy. In mapping advertisements to the hierarchy, the system parses the contents of real-estate advertisements in an HTML file, and extracts object information based on the values of the predefined attributes for the advertisements. Then, the system maps objects to the hierarchy based on the extracted information. The hierarchy becomes the main navigation space. A user can also navigate in a conceptual space by combining two scales in a nested line diagram.

4.4.5. Proposed Approach

Incremental Development
The main difference between the previous work and our work is an emphasis on incremental development and evolution for a document management and retrieval system (Kim and Compton 2001b; 2001c). The main aim of our study is a browsing mechanism for retrieval which can be collaboratively created and maintained and where users evolve their own organisation of documents but are assisted in this to facilitate improvement of the search performance of the system as it evolves.

Web-based System
Another difference is that our focus is on a Web-based system (Kim and Compton 2000; 2001a) using a hypertext representation of the links to a node, but without a graphical display of the overall lattice. Lin (1997) discussed how visualisation through a graphical interface could enhance information retrieval. In fact, except for the Godin et al.
method, all navigation mechanisms in the applications of FCA are devoted to exploring the lattice graph itself. Figure 4.6 shows a typical navigation space in the FCA approach. Even though we agree that the lattice diagram can be a useful tool to analyse and explore the whole map of a domain, we anticipate that most Web users are unfamiliar or uncomfortable with concept lattice diagrams and viewing of the whole lattice diagram will also remain a problem. Accordingly we have developed a Web-based lattice display using hyperlinks and URLs. We believe that the hyperlink technique is a fairly natural simplification for a lattice display without losing any advantage of FCA. It is also very comprehensive and natural to use for Web users. The browsing system developed by Cole and Eklund (2001) is also Web-based, but it is implemented as a stand-alone application using line diagrams and it can only browse the lattice with pre-defined scales.53

![Figure 4.6. An example of a line diagram (extracted from Groh et al. 1998).](image)

Integration with General Information Retrieval Mechanisms

We have also integrated a number of information retrieval mechanisms into lattice browsing. Firstly, a Boolean query interface is combined with the FCA browsing interface in a similar way to Carpineto and Romano (1998). The approach of Carpineto and Romano is simply to move to a relevant portion of the lattice with a user’s query. In our approach, a number of information retrieval techniques are combined to the query interface such as eliminating stopwords, stemming and expanding user query based on synonyms and abbreviations. In other words, a user can formulate a query by entering any textwords as a conventional Boolean query interface. Then, the system normalises the user query by eliminating stopwords and stemming, and extends the query based on abbreviations and synonyms. Following that, the system identifies the most relevant portion in the lattice for the query. The user can navigate the relevant documents starting with the portion of the lattice.

Secondly, a textword search is supported. This is invoked automatically to identify the relevant documents from the context when the system fails to get a result from the lattice nodes. The systemformulates a sub-lattice with the results which contain the user’s query in their context (a set of documents) and their keywords (a set of keywords). Navigation can be carried out on this sub-lattice.

Conceptual Scaling

Conceptual scaling is also supported to allow users to get more specific results and to search relevant documents by a relationship between the domain attributes and the keywords of documents. One might consider that conceptual scaling is similar to the optimising process of a rule tree described in Chapter3. However, in essence, conceptual scaling gives a view of a lattice formed from objects that have the specified attribute value pairs. On the other hand, the tree optimisation is to optimise the rule tree to reduce the number of conditions to be reviewed by the user, and the number of interactions between the user and the system.

In the proposed approach, a many-valued context with the obvious attributes for the evolved domain is formulated and relevant values in a one-valued context of the
keyword sets are grouped (Kim and Compton 2001a; 2001b). A nested structure is then automatically derived on the fly from the attributes of the many-valued context and the grouping names of the one-valued context corresponding to search results. That is, a concept lattice is built using the keyword sets of the resulting documents in response to a user’s query as an outer structure and from this a nested structure is produced. A user can navigate recursively among the nested attributes in regard to the interrelationship between the outer structure (keywords) and the nested attributes.

The e-mail management system and real estate system of Cole et al. (Cole and Stumme 2000; Cole and Eklund 2001) also support conceptual scaling. The techniques of conceptual scaling of Cole et al. and our approach are quite similar. In the system of e-mail management (CEM), scaling is applied to a one-valued context for the attribute catchword. As indicated earlier, a user can define a hierarchy on more general catchwords and e-mails are managed based on this hierarchy. A cluster in the hierarchy can be a scale (i.e., default scale). In addition, the user can establish scales to group together related catchwords. Then, the user begins their search by requesting a scale they have defined or a default scale in a single line diagram. The user can also navigate e-mails by combining two scales in a nested line diagram.

In the real estate system (Cole and Eklund 2001), attributes and their values for the advertisements are pre-defined, and are ordered in a hierarchy. Objects and their attributes are managed in a many-valued context, and the objects are mapped into the hierarchy. Then, the hierarchy becomes the main navigation and serves as a general hierarchical classification system. Scales are predefined for each attribute and its values in the many-valued context. A user can navigate in a conceptual space by combining two scales in a nested line diagram. For example, if the user is interested in observing fully-furnished mid-range properties, they can combine the scale “price” with a scale for “furnished” in the conceptual space.

On the contrary, in our approach, a concept lattice is dynamically built with the annotated documents and their keywords as an outer structure. This outer lattice structure along with the keywords set becomes the main navigation space. Then, a
many-valued context is defined with attributes for the evolving domain based on a partially ordered hierarchy among the attributes. The hierarchy can be considered as an ontological structure of the evolving domain described with the most obvious attribute. Each attribute and its values in the many-valued context become a scale. A knowledge engineer (or a user) can also group relevant values in the keywords of documents and a grouping becomes a scale. The groupings can be defined whenever they are required. A nested structure then is constructed dynamically and automatically from the search results of a corresponding concept of the outer lattice at run time. That is, all relevant scales for the search results are extracted from the scales of the many-valued context and the groupings of the one-valued context, and are included in a nested structure using pop-up and pull-down menus. A menu structure is incorporated with the hierarchy of the many-valued context and/or the hierarchy of groupings. The user can navigate recursively among the nested attributes by observing the interrelationship between the attributes as well as the outer structure. The system supports a link to the nested structure of a concept of the outer lattice.

The main difference between the approach of Cole et al. and the proposed approach is that the systems of Cole et al. start with a predefined hierarchy which transforms into a basic structure of managing objects and a main navigation space. On the other hand, in the proposed approach a lattice structure develops automatically as the system evolves. This lattice structure is driven from the annotated keywords for documents, and the lattice evolves into a basic structure of indexing documents and a main browsing space. Another difference is that the systems of Cole et al. display a selected scale in a line diagram and combine two scales using a nested line diagram in a conceptual space. Whereas, the proposed system includes all scales which are relevant to a search result at run time, in pop-up and pull-down menus as a nested structure of the outer structure.

Finally, the systems of Cole et al. are implemented as a stand-alone application, whereas our focus is both for a single user or a multi-user application focusing on Web environments and Web users organising objects identified by their URL. The browsing system developed by Cole and Eklund (2001) is also Web-based, in that it gets material from the Web and it can only browse the lattice with pre-defined scales. However, the
advantages of both methods can depend on the properties of the applied domains. The approach of Cole et al. can be a good choice where an ontology can be imported or can be easily constructed for the application. In addition, their graphical user interface (a line diagram) can be useful for a reasonably small domain.

4.5. Chapter Summary

This chapter presented an overview of the basic theories of Formal Concept Analysis and its application areas especially focusing on the field of information retrieval. Algorithms for computing all concepts of a formal context and its concept lattice were surveyed. A variety of algorithms exist in the literature. There is no “best” algorithm, rather the performance of algorithms depends on the properties of input data such as the size and density of contexts (Kuznetsov and Ob’edkov 2001). The main issues for computing all the formal concepts of a context are related to how to generate all the concepts of a context and how to avoid repetitive generation of the same concept. Conceptual scaling was introduced in order to deal with many-valued attributes as well as to reduce the complexity of visualisation in one-valued contexts. It was successfully demonstrated in TOSCANA. The resulting conceptual hierarchy allows users to have a structured overview over their queries, and to interpret the concepts based on the interrelationship between the attributes.

The method of Formal Concept Analysis was applied to a wide range of application fields. Lattice-based information retrieval using FCA is one of those areas. Its significant advantage for information retrieval is that the mathematical formulae of FCA can produce a conceptual structure which provides all possible generalisation and specialisation relations among the concept nodes so that it can be used as a browsing scheme. Lattice-based models for information retrieval in the literature were reviewed by addressing similarities and differences with the approach that we propose. The main difference between the previous work and our work is an emphasis on incremental development and evolution, and knowledge acquisition tools to support these for domain-specific document retrieval systems. A further difference is that our focus is on a Web-based system managing documents distributed across the Web. The details of the proposed approach will be given in Chapter 5.
Chapter 5
A Formal Framework of
Document Management and Retrieval for Specialised Domains

This chapter presents a theoretical framework for a domain-specific document management and retrieval system that we propose. This is based on Formal Concept Analysis (FCA) and is aimed for a Web-based system for organisations in specialised domains. This approach allows users themselves to freely annotate their documents and to find appropriate annotations for new documents. Any relevant documents can be managed by annotating with any terms the users or authors prefer. This results in the automatic generation of a browsing system which can fit into a predetermined taxonomical ontology used for browsing in information retrieval.

The main focus is on incremental development and evolution, and we provide knowledge acquisition tools to support this. The knowledge acquisition mechanisms encourage reuse of terms used by others and imported terms from other taxonomies. A conceptual lattice-based browsing structure for retrieval is automatically and incrementally created and maintained from the annotation of users. Document retrieval is based on navigating this lattice structure. The browsing structure is scaled (conceptual scaling) with the evolving ontological structure of the domain to allow more specific results or to group relevant documents together. We have previously described the main features of this system (Kim and Compton 2000; 2001a; 2001b; 2001c).

Section 5.1 defines the basic notions of the system such as formal contexts, formal concepts and concept lattice. The definitions and formulas in this chapter closely adhere to the basic work of Ganter and Wille (1999). Here, the word context and concept are used to respectively mean a formal context and a formal concept as in Chapter 4. Section 5.2 introduces an incremental algorithm we have developed for building a concept lattice. Section 5.3 presents how documents can be managed by users
themselves cooperating with the knowledge acquisition tools we propose. Section 5.4 describes document retrieval in the proposed approach using both browsing (of the concept lattice) and a Boolean query interface. Section 5.5 presents conceptual scaling both in a many-valued context and a one-valued context.

5.1. Basic Notions of the System

5.1.1. Formal Context

The most basic data structure of Formal Concept Analysis is a formal context. In the original formulation of FCA, an object was implicitly assumed to have some sort of unity or identity so that the attributes applied to the whole object (e.g., a dog has four legs). Clearly documents do not have the sort of unity where attributes will necessarily apply to the whole document. Any sort of keyword or attribute approach to document management has the same problem. As well it is imagined that many documents will increasingly be structured with URLs having multiple sections. However in order to use FCA, it is assumed that documents correspond to objects and the keywords or terms attached to documents by a user constitute attribute sets. A formal context is defined for the system that we propose as follows:

**Definition 1**: A document-based *formal context* is a triple \( C = (D, K, I) \) where \( D \) is a set of documents (objects), \( K \) is a set of keywords (attributes) and \( I \) is a binary relation which indicates whether \( k \) is a keyword of a document \( d \). If \( k \) is a keyword of \( d \), it is written \( dk \) or \( (d, k) \in I \).

A context is represented by a cross table with the document names in the rows and the keyword names in the columns. Table 5.1 shows an example of the formal context of \( C \) where a set of documents \( D \) is \{1, 2, 3, 4, 5\}\(^{54}\), a set of keywords \( K \) is \{artificial intelligence, knowledge acquisition, machine learning, behavioural cloning, knowledge engineering, knowledge representation, belief revision, ontology\} and the relation \( I \) is

\(^{54}\) Here, numbers are used to indicate document names or URLs for reasons of convenience.
Table 5.1. A part of the formal context in the proposed system.

<table>
<thead>
<tr>
<th></th>
<th>Artificial Intelligence</th>
<th>Knowledge Acquisition</th>
<th>Machine Learning</th>
<th>Behavioural Cloning</th>
<th>Knowledge Engineering</th>
<th>Knowledge Representation</th>
<th>Belief Revision</th>
<th>Ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

A symbol “X” designates that a particular document has the corresponding keywords.

\{(1, artificial intelligence), (1, knowledge acquisition), (1, machine learning), (1, behavioural cloning), (2, artificial intelligence), ..., (4, knowledge representation), (4, belief revision), (5, artificial intelligence), (5, knowledge acquisition), (5, knowledge representation), (5, ontology)\}.

5.1.2. Formal Concept

Formal concepts reflect the relationships between documents and keywords. A formal concept for the proposed system is defined as follows:

**Definition 2:** A formal concept of the context \( C = (D, K, I) \) is defined as a pair \((X, Y)\) such that \( X \subseteq D, Y \subseteq K, X' = Y \) and \( Y' = X \) where \( X \mapsto X' := \{ k \in K \mid \forall d \in X: (d, k) \in I \} \) and \( Y \mapsto Y' := \{ d \in D \mid \forall k \in Y: (d, k) \in I \} \). \( X \) is called the extent and \( Y \) is called the intent of the concept \((X, Y)\).

To construct a conceptual structure, it is necessary to find all formal concepts of the context \( C \). The following formula is used to construct all concepts of \( C \):

\[
X' = \bigcap_{d \in X} \{d\}'
\]

First, all extents \( X' \) for all intents \( \{d\}' \) with \( d \in D \) is determined. Then the intents of all determined extents are computed. The set of all formal concepts of \( C \) is designated by \( \mathcal{B}(D, K, I) \). \( \mathcal{B}(C) \) denotes the shortened form of \( \mathcal{B}(D, K, I) \).
5.1.3. Concept Lattice

The concept lattice is the conceptual structure of FCA. To build a concept lattice it is necessary to find the subconcept-superconcept relationships between all the formal concepts $\mathscr{B}(D, K, I)$. This is formalised by $(X_1, Y_1) \leq (X_2, Y_2) \iff X_1 \subseteq X_2 (\iff Y_2 \subseteq Y_1)$ where $(X_1, Y_1)$ is called a subconcept of $(X_2, Y_2)$ and $(X_2, Y_2)$ is called a superconcept of $(X_1, Y_1)$. The relation $\leq$ is called the hierarchical order of the concepts. The set of all formal concepts of the context ordered by this subconcept-superconcept relationship, is called the concept lattice of the context $C$ denoted by $\mathcal{L}(D, K, I)$.

The line diagram in Figure 5.1 shows the concept lattice of $C$ in Table 5.1. Each node represents a formal concept $(X, Y)$ where $X$ is the set of documents and $Y$ is the set of keywords. In the proposed application, this structure is reformulated incrementally and automatically by the addition of a new document with a set of keywords or by refining the existing keywords of the documents. A more detailed explanation will be given in the following section.

![Figure 5.1. A concept lattice of the formal context $C$ in Table 5.1.](image-url)
5.2. Incremental Construction of a Concept Lattice

Incremental methods are used to generate a concept lattice starting from a single document with its keywords set. The concept lattice is updated whenever a new document is added with a set of keywords or the keywords of existing documents are refined. The incremental algorithms in the literature focus on adding a new object into the lattice. However, in the proposed application, users can refine the set of keywords of their documents at any time if they desire. As a consequence, we could not directly use the algorithms in the literature so that we chose to develop further incremental algorithms to construct a concept lattice for our specific situation. However, we are not aiming to prove correctness of the algorithms and are not making any claims of greater efficiency that other algorithms, but the algorithms present a detailed description of the implemented approach. For proofs of correctness of incremental algorithms refer to Godin et al. (1994). Note that the study of Godin et al. (1994) only addressed the cases of adding concepts, not for refining concepts already in the concept lattice of FCA.

5.2.1. Basic Definitions of the Algorithms

It is supposed that the existing formal context \( C = (D, K, I) \) where \( D \) is a set of documents, \( K \) is a set of keywords and \( I \) is a binary relation between \( D \) and \( K \). Recall that a formal concept of a context \( C \) is a pair \( (X, Y) \) where \( X \) is the extent and \( Y \) is the intent of the concept. The set of all formal concepts and the concept lattice of \((D, K, I)\) are denoted as \( \mathcal{B}(C) \) and \( \mathcal{L}(C) \), respectively. Now, let \( \text{ext}(\mathcal{B}(C)) \) be the set of all extents and \( \text{int}(\mathcal{B}(C)) \) be the set of all intents of \( \mathcal{B}(C) \), a revised formal context of \( C \) is defined as follows for adding a document and refining the keyword set of an existing document.

**Definition 3:** Let \( C = (D, K, I) \) be a formal context, \( \delta \) be a document and \( \Gamma \) be the set of keywords of \( \delta \). For adding a new document \( \delta \ (\notin D) \), the revised formal context of \( C \) is defined as \( C^+ = (D^+, K^+, I^+) \) where \( D^+ = D \cup \{\delta\} \), \( K^+ = K \cup \Gamma \) and \( I^+ = I \cup \{\langle \delta, k \rangle \mid k \in \Gamma\} \}. \) In the case of refining the keywords of an existing document \( \delta \ (\in D) \), the set of document \( D \) remains unchanged \( (D^+ = D) \). The set of keywords \( K^+ = K \cup \Gamma \setminus \{k \in K \mid \text{there does not exist a } d \in D \text{ such that } (d \neq \delta) \text{ and } (d, k) \in I\} \). \( I^+ \) is \( I \setminus ((\delta, k) \in I \mid k \in K) \) \( \cup \{\langle \delta, k \rangle \mid k \in \Gamma\} \).
5.2.2. Description of the Algorithms

Algorithm 1 describes the incremental algorithm for adding a new document \( \delta \) with a set of keywords \( \Gamma \). Firstly, all possible new concepts from the new case (the document and its keywords) are computed using the procedure \( \text{computeNewConcepts} \). When the procedure \( \text{computeNewConcepts} \) is completed, all possible formal concepts of the revised context \( C^+ \) are computed resulting in \( \mathfrak{B}(C^+) \).

Secondly, the procedure \( \text{reconstructLattice} \) (Algorithm 2) is performed to reformulate the subconcept and superconcept relationships for all formal concepts whose intent includes at least an element of \( \Gamma \) of the new document \( \delta \). This results in a new lattice \( \mathfrak{L}(D^+, K^+, I^+) \) of \( C^+ \).

```
Input: \( C^+ = (D^+, K^+, I^+) \) - The revised context of \((D, K, I)\)  
\( \mathfrak{L}(C) \) - The concept lattice of \((D, K, I)\)  
\( \Gamma \) - The set of keywords (of the new document \( \delta \))  
Output: \( \mathfrak{L}(C^+) \) - The concept lattice of the revised context \( C^+ = (D^+, K^+, I^+) \)

Procedure \( \text{addDocument}(C^+, \mathfrak{L}(C), \Gamma) \)  
1 Begin  
2 \( \mathfrak{B}(C) \leftarrow \) the set of all concepts consist in \( \mathfrak{L}(C) \);  
3 \( \mathfrak{B}(C^+) \leftarrow \text{computeNewConcepts}(C^+, \mathfrak{B}(C), \Gamma); \)  
4 \( \mathfrak{L}(C^+) \leftarrow \text{reconstructLattice}(\mathfrak{L}(C), \mathfrak{B}(C^+), \Gamma); \)  
5 Return \( \mathfrak{L}(C^+) \);  
6 End

Algorithm 1. The algorithm for adding a new document.
```

```
Input: \( \mathfrak{L} \) - A concept lattice  
\( \mathfrak{I} \) - A set of concepts  
\( \Gamma \) - A set of keywords  
Output: \( \mathfrak{L}^+ \) - The revised concept lattice of \( \mathfrak{L} \)

Procedure \( \text{reconstructLattice}(\mathfrak{L}, \mathfrak{I}, \Gamma) \)  
1 Begin  
2 For each formal concept \((X, Y) \in \mathfrak{I} \) do  
3 If \( Y \cap \Gamma \neq \emptyset \) then  
4 \( \mathfrak{L}^+ \leftarrow \) Reconstruct the superconcepts and subconcepts of the concept \((X, Y) \) in \( \mathfrak{L} \);  
5 End if  
6 End for  
7 Return \( \mathfrak{L}^+ \);  
8 End

Algorithm 2. Reconstruction of the concept lattice.
```
In the procedure \textit{computeNewConcepts} (Algorithm 3), it is started by formulating a pair (X, Y) where Y is \(\Gamma\) and X is the document set associated with \(\Gamma\) of the revised context \(C^*\). The procedure \textit{addOneConcept} is then performed to determine whether (X, Y) is a new concept of \(C^*\). Secondly, the following process is applied for each element \(\gamma\) of \(\Gamma\). A pair \((X_1, Y_1)\) is constructed where \(X_1\) is the set of documents which is associated with the element \(\gamma\) and \(Y_1\) is the set of keywords associated with \(X_1\). Then, the procedure \textit{addOneConcept} is performed to determine whether the concept \((X_1, Y_1)\) can be a new concept of the revised context \(C^*\). Next, the intersection of \(X_1\) with the \textit{extent} sets of \(\mathcal{S}^+\) is obtained by the definition \(\text{intersect}(X_1) = \{X_1 \cap E \mid E \in \text{ext}(\mathcal{S}^+)\} \setminus \text{ext}(\mathcal{S}^+)\). Then, for each element \(X_2 \in \text{intersect}(X_1)\) satisfying the condition: \(X_2 \notin \text{ext}(\mathcal{S}^+)\), a concept \((X_2, Y_2)\) is formulated where \(Y_2\) is the set of keywords associated with \(X_2\), and the procedure \textit{addOneConcept} is performed for the concept \((X_2, Y_2)\).

To incorporate a given pair \((X, Y)\) into the concepts of \(C^*\) in the procedure \textit{addOneConcept} (Algorithm 4), it is determined whether \(Y\) is an intent of the given set of concepts \(\mathcal{S}\). If \(Y\) is a member of \(\text{int}(\mathcal{S})\), then the extent \(X'\) is obtained where \((X', Y) \in \mathcal{S}\). If the cardinality of \(X'\) is less than the cardinality of \(X\), the existing concept \((X', Y)\)

---

**Input:**
\[C^* = (D^*, K^*, \Gamma^*)\] - The revised context of \(C = (D, K, \Gamma)\)
\[\mathcal{S} - \text{The set of concepts of } C\]
\[\Gamma^* - \text{The set of keywords (of the new document } \delta)\]

**Output:**
\[\mathcal{S}^+ - \text{The set of concepts of } C^+\]

**Procedure** \textit{computeNewConcepts}(\(C^*, \mathcal{S}, \Gamma\))

1. \textbf{Begin}
2. \(Y \leftarrow \Gamma; X \leftarrow \{d \in D^* \mid d^\gamma k \text{ for all } k \in Y\};\)
3. \(\mathcal{S}^+ \leftarrow \text{addOneConcept}(\mathcal{S}, (X, Y));\)
4. \textbf{For each } \gamma \in \Gamma \textbf{ do}
5. \(X_1 \leftarrow \{d \in D^* \mid d^\gamma \gamma\}; Y_1 \leftarrow \{k \in K^* \mid d^\gamma k \text{ for all } d \in X_1\};\)
6. \(\mathcal{S}^+ \leftarrow \text{addOneConcept}(\mathcal{S}^+, (X_1, Y_1));\)
7. \(\text{intersect}(X_1) = \{X_1 \cap E \mid E \in \text{ext}(\mathcal{S}^+)\} \setminus \text{ext}(\mathcal{S}^+);\)
8. \textbf{For each } \gamma \in \text{intersect}(X_1) \textbf{ do}
9. \(Y_2 \leftarrow \{k \in K^* \mid d^\gamma k \text{ for all } d \in X_2\};\)
10. \(\mathcal{S}^+ \leftarrow \text{addOneConcept}(\mathcal{S}^+, (X_2, Y_2));\)
11. \textbf{End for}
12. \textbf{End for}
13. \textbf{Return } \mathcal{S}^+;\)
14. \textbf{End}

---

**Algorithm 3.** Construction of concepts connected with the new document.
Input: $\mathcal{I}$ - A set of concepts
(X, Y) - A concept where X is the extent and Y is the intent of the concept (X, Y)

Output: $\mathcal{I}^+$ - A revised set of concepts of $\mathcal{I}$

Procedure addOneConcept ($\mathcal{I}$, (X, Y))

1. Begin
2. $\mathcal{I}^* \leftarrow \mathcal{I}$;
3. If $X \neq \emptyset$ then
4. If $Y \in \text{int}(\mathcal{I})$ then
5. Let $X' \in \text{ext}(\mathcal{I})$ such that $(X', Y) \in \mathcal{I}$;
6. If $|X'| < |X|$ then
7. $\mathcal{I}^* \leftarrow \mathcal{I} \setminus \{(X', Y)\} \cup \{(X, Y)\}$;
8. End if
9. Else
10. $\mathcal{I}^* \leftarrow \mathcal{I} \cup \{(X, Y)\}$;
11. End if
12. End if
13. Return $\mathcal{I}^*$;
14. End

Algorithm 4. Insertion of a new concept into the set of all concepts.

is eliminated from $\mathcal{I}$ and the given concept (X, Y) is added into $\mathcal{I}$. Otherwise, the new concept (X, Y) is not cooperated into $\mathcal{I}$. When Y is not a member of $\text{int}(\mathcal{I})$, the concept (X, Y) is just appended as a member of $\mathcal{I}$.

Algorithm 5 describes the procedure for refining the keyword set of an existing document. The main difference between Algorithm 5 (refining an existing document) and Algorithm 1 (adding a new document) comes from the deleted keyword set of the refined document $\delta$ (see difference between Algorithm 3 and 6). Algorithm 6 ($\text{refineConcepts}$) is started by computing the keyword set deleted from $\delta$ (Line 2: $\Gamma_2 \leftarrow \Gamma_1 \setminus \Gamma$). If the set is empty, the rest of Algorithm 6 is essentially the same as Algorithm 3. If the deleted set is not empty, then Algorithm 7 is performed first to refine the concepts which include the deleted keywords. The concepts are eliminated or revised according to the cardinality of their extent.

55 In the Basic Theorem of FCA the concept ($\emptyset$, {all attributes}) is added as an element of the given concepts set. However, in our implementation, the node of the concept ($\emptyset$, {all attributes}) is added when the lattice is constructed only if it is necessary. Thus, we do not consider the case that its extent is empty.
Algorithm 5. The algorithm for refining an existing document.

```
Input: \( C^* = (D^*, K^*, I^*) \) - The revised context of \( (D, K, I) \)
\( \mathcal{E}(C) \) - The concept lattice of \( (D, K, I) \)
\( \delta \) - A refined document
\( \Gamma \) - The new set of keywords of \( \delta \)
\( \Gamma_1 \) - The old set of keywords of \( \delta \)
Output: \( \mathcal{E}(C^*) \) - The lattice of the revised context \( C^* = (D^*, K^*, I^*) \)

Procedure refineDocument(C*, \( \mathcal{E}(C) \), \( \delta \), \( \Gamma \), \( \Gamma_1 \) )
1 Begin
2 \( \mathfrak{A}(C) \leftarrow \) the set of all concepts consist in \( \mathcal{E}(C) \);
3 \( \mathfrak{A}(C^*) \leftarrow \text{refineConcepts}(C^*, \mathfrak{A}(C), \delta, \Gamma, \Gamma_1) \);
4 \( \mathcal{E}(C^*) \leftarrow \text{reconstructLattice}(\mathcal{E}(C), \mathfrak{A}(C^*), \Gamma \cup \Gamma_1) \);
5 \text{Return} \( \mathcal{E}(C^*) \);
6 End
```

Algorithm 6. Refinement of concepts connected with the refined document.

```
Input: \( C^* = (D^*, K^*, I^*) \) - A revised context of \( C = (D, K, I) \)
\( \mathfrak{S} \) - A set of concepts
\( \mathfrak{g} \) - The refined document
\( \Gamma \) - The new set of keywords of \( \mathfrak{g} \)
\( \Gamma_1 \) - The old set of keywords of \( \mathfrak{g} \)
Output: \( \mathfrak{S}^* \) - A refined set of concepts of \( \mathfrak{S} \)

Procedure refineConcepts(C*, \( \mathfrak{S} \), \( \mathfrak{g} \), \( \Gamma \), \( \Gamma_1 \) )
1 Begin
2 \( \mathfrak{S}^* \leftarrow \mathfrak{S} \);
3 \( \Gamma_2 \leftarrow \Gamma_1 \setminus \Gamma \);
4 If \( \Gamma_2 \neq \emptyset \) then
5 \( \mathfrak{S}^* \leftarrow \text{refineConceptsForDeletedTerms}(\mathfrak{S}^*, \mathfrak{g}, \Gamma_2) \);
6 Endif
7 \( Y \leftarrow \Gamma; X \leftarrow \{d \in D^* \mid d^* \in Y \} \);
8 \( \mathfrak{S}^* \leftarrow \text{addOneConcept}(\mathfrak{S}^*, (X, Y)) \);
9 \( \Gamma_3 \leftarrow \Gamma \setminus \Gamma_1 \);
10 For each \( \gamma \in \Gamma_3 \) do
11 \( X_1 \leftarrow \{d \in D^* \mid d^* \gamma \}; Y_1 \leftarrow \{k \in K^* \mid d^* \gamma \} \) for all \( k \in X_1 \);
12 \( \mathfrak{S}^* \leftarrow \text{addOneConcept}(\mathfrak{S}^*, (X_1, Y_1)) \);
13 intersect(X_1) = \{X_1 \cap E \mid E \in \text{ext}(\mathfrak{S}^*) \} \setminus \text{ext}(\mathfrak{S}^*) \);
14 For each \( X_2 \in \text{intersect}(X_1) \) do
15 \( Y_2 \leftarrow \{k \in K^* \mid d^* \gamma \} \) for all \( d \in X_2 \);
16 \( \mathfrak{S}^* \leftarrow \text{addOneConcept}(\mathfrak{S}^*, (X_2, Y_2)) \);
17 End for
18 End for
19 \text{Return} \( \mathfrak{S}^* \);
20 End
```
Input: \( \mathcal{I} \) - A set of concepts
\( \delta \) - The refined document
\( \Gamma_2 \) - A set of keywords

Output: \( \mathcal{I}^* \) - A refined set of concepts of \( \mathcal{I} \)

Procedure refineConceptsForDeletedTerms(\( \mathcal{I}, \delta, \Gamma_2 \))

1 Begin
2 \( \mathcal{I}^* \leftarrow \mathcal{I} ; \)
3 For each concept \((X, Y) \in \mathcal{I}\)
4   If \( Y \cap \Gamma_2 \neq \emptyset \) & \( \delta \in X \) then
5     If \( |X| = 1 \) then
6        \( Y^' \leftarrow Y \setminus \Gamma_2 ; \)
7        \( \mathcal{I}^* \leftarrow \mathcal{I}^* \setminus \{(X, Y) \cup \{(X, Y')\}; \)
8     Else if \( |X| > 1 \) then
9        \( X^' \leftarrow X \setminus \{\delta\}; \)
10       If \( X^' \notin \text{ext}(\mathcal{I}) \) then
11          \( \mathcal{I}^* \leftarrow \mathcal{I}^* \setminus \{(X, Y) \cup \{(X', Y)\}; \)
12        Else
13          \( \mathcal{I}^* \leftarrow \mathcal{I}^* \setminus (X, Y); \)
14       End if
15     End if
16   End for
17 Return \( \mathcal{I}^* ; \)
18 End

Algorithm 7. Refinement of the concepts which include the keywords deleted.

We briefly present the worst case time complexity for the algorithms. In the worst case, \( addOneConcept() \) is \( O(|\mathcal{I}|) \) where \( |\mathcal{I}| \) denotes the number of formal concepts (i.e., the size of the concept lattice). However, as \( \text{int}(\mathcal{I}) \) is implemented by a hash table mapping the keys from \( \text{int}(\mathcal{I}) \) to the values in \( \text{ext}(\mathcal{I}) \), on average, this function is likely to be more efficient, i.e., \( O(1) \). There is also a tradeoff between time and space costs. The worst case complexity for \( \text{computeNewConcepts()} \) is \( O(|\mathcal{I}|^2|\Gamma|) \) where \( |\Gamma| \) is the number of keywords of the new document. Therefore, the time complexity, in the worst case, for adding a new document - \( addDocument() \) - is \( O(|\mathcal{I}|^2|\Gamma|) \). In practice, the complexity of \( addDocument() \) is approximately estimated at \( O(|\mathcal{I}|^2|\Gamma|) \) as the complexity of \( addOneConcept() \), on average, is \( O(1) \). For refining an existing document: the time complexity of function \( refineDocument() \), in the worst case, is \( O(|\mathcal{I}|^4|\Gamma|) \) because there is another \textbf{For} loop to refine the concepts which include the deleted keywords of a refined document. However, on average, this complexity can be reduced to \( O(|\mathcal{I}|^4|\Gamma|) \) for the use of hash table in our implementation. Note that the complexity of incremental algorithms given in Chapter 4 is for adding a new object.
5.3. Document Management

In this proposed approach, users themselves evolve their own organisation of documents based on the free annotation of their documents with a set of keywords. They can also refine the keywords of existing documents at will. When a user assigns keywords to a document, some keywords may be missed and/or may be prompted by stored documents or domain knowledge. Thus, it is appropriate to have certain knowledge acquisition mechanisms to improve the search performance of the system as it evolves. Figure 5.2 shows the overview of the annotation process of keywords for a document.

![Diagram](image.png)

**Figure 5.2.** The annotation process of keywords for a document.
A number of knowledge acquisition techniques are used to suggest possible annotations during this annotation process. The knowledge acquisition for conceptual modelling has been a fairly centralised process in most expert system applications. In contrast, our system guides the users (or annotators) to capture missed concepts by suggesting possible keywords, rather than an expert. The annotation process is divided into a number of phases. Each phase is presented in detail in the following sections.

5.3.1. Phase One: Reusing Terms in the System

The system is expected to be collaboratively developed and maintained over time in Web-based distributed environments by authors (users). Hence, it is crucial to guide them to use controlled vocabularies, if they exist, to help ensure consistency of the system. This is accomplished by reusing and sharing terms already in the system.

When a user annotates their document with a set of keywords, the user is provided with a list of keywords already available in the system that have been added by others in alphabetical order. The system also provides a list of keywords based on each of annotators. The user can select keywords from them or can enter further text words which in turn will be available to future users. The system also displays domain terms from imported taxonomies in a list combined with the terms used by others.

5.3.2. Phase Two: Using Imported Terms from Taxonomies

Information retrieval often incorporates the use of thesauri or taxonomies as background knowledge to extend a user’s query. A number of researchers (Carpineto and Romano 1996a; Cole and Eklund 1996a; Priss 2000a) have utilised a domain thesaurus for their lattice retrieval processes and presented experimental evidence that adding a thesaurus to a concept lattice improves its retrieval performance.

However, there is little point in considering inheritance as part of a reasoning mechanism with taxonomies. This is because commonly available taxonomies, thesauri or classifications are too general. So that when such a taxonomy is applied to a
particular domain, the inheritance between the terms in the hierarchy of the taxonomy are not often transitive.

To observe the situation, we start with the subsumption hierarchy of a taxonomy which consists of a partially ordered set \((K, \leq)\) and a context \((D, K, I)\). The context assigns the related taxonomy terms \((\in K)\) to the documents in the set \(D\). \(K\) is the set of terms in the taxonomy. When the taxonomy is involved in an information retrieval process, the following compatibility condition\(^5^6\) is assumed for a subsumption hierarchy.

\[
\forall d \in D, \ k, l \in K: \ (d, k) \in I, \ k \leq l \Rightarrow (d, l) \in I
\]

The problem is that this compatibility condition is not always inheritable when the taxonomy is used for a specific application domain. For example, suppose there is a document \(d \in D\) and two terms (logic \(<\) knowledge representation) \(\in K\) from Figure 5.3 (a), and assume \(d\) is associated with the term \(logic\) ((\(d, \ logic\) \(\in I\)). But \(d\) may or may not be relevant to the term \(knowledge\) \(representation\) ((\(d, \ knowledge\) \(representation\) \(\in I\)) or ((\(d, \ knowledge\) \(representation\) \(\notin I\))). In other words, logic is a typical method for knowledge representation and reasoning. Therefore, \(d\) can be only relevant to some reasoning mechanisms using logic, but not knowledge representation techniques or vice versa.

For another example, assume there is a document \(d \in D\) and two terms (data mining \(<\) artificial intelligence) from Figure 5.3 (c), and suppose \(d\) is related to the term \(data\) \(mining\) ((\(d, \ data\) \(mining\) \(\in I\)). But the term \(artificial\) \(intelligence\) may or may not be coupled to the document \(d\). Rather \(d\) might be related to data mining with database techniques to discover relationships between data items or association rules from databases. Therefore, reasoning by transitivity does not always improve retrieval performance and can even give poor results (Nikolai 1999).

---

\(^{56}\) Stumme (1999) and Nikolai (1999) describe this problem in their papers. We will explain the problem following the compatibility notion of Stumme. Note that this situation will be different in a well-defined ontological taxonomy which is built to hold the semantics of an “is-a” relation (Guarino and Welty 2000).
Figure 5.3. Examples of hierarchies extracted from taxonomies.

(a) ACM Computing Classification System\textsuperscript{57}.

(b) ASIS&T thesaurus for Information Science\textsuperscript{58}. (c) Open Directory Project\textsuperscript{59}.

However, thesauri or taxonomies have significant implications for understanding, sharing, reuse and integration. The use of these hence is common as background knowledge of a reasoning mechanism.

The only browsing mechanism we propose is FCA so there is little point in considering a reasoning mechanism. Thus, we also import a number of taxonomies as others do, but we try to solve the compatibility problem described above. Consequently, we import only the domain terms from the relative thesauri or taxonomies to the evolving domain, but the inheritance between the terms of an ordered set remains on the conceptual structure of FCA. In other words, in our approach the value is in suggesting all parents of the term in a list, rather than the hierarchy itself. The inheritance between the terms of ordered sets is a question for knowledge acquisition and remains on the conceptual structure of FCA.

\textsuperscript{57} ASIS&T (American Society for Information Science and Technology) - http://www.asis.org/Publications/Thesaurus/isframe.htm (2002).


\textsuperscript{59} http://dmoz.org (2002).
When a new document is added and the user enters a term that occurs in the taxonomy all parents of the term up the hierarchy (i.e., predecessors) are also displayed. Any of these terms can be selected by the user and added to the document as keywords. This means that the various superclasses are considered as Boolean attributes. The user can judge whether the suggested terms are applicable to their document, and is free to select none, one or some of these terms without considering any hierarchies of the terms. Then, FCA constructs a concept lattice which holds inheritance between terms in the lattice hierarchies in relation to documents.

There are good reasons for having taxonomies, but the initial motivation for developing mechanisms for combining both inheritance and heuristic reasoning was probably to avoid a user having to enter the entire taxonomy. For example, if the user had already entered “dog” it would be inappropriate to ask them to enter “mammal” and “animal” as well to allow the appropriate rules to fire. Here however we are not asking the end-user browsing the system to enter such terms, but the user (annotator) who is entering a document, and the hierarchy is used to suggest which terms may be relevant.

Figure 5.3 shows partial examples of the taxonomies we import. The ACM computing classification system and ASIS&T thesaurus for Information Science have been imported from commonly available Web sites. A hybrid called UNSW has been developed based on the Open Directory Project, the KA2 community Web60, and the research areas at the School of Computer Science and Engineering61, UNSW.

The reason for importing from a number of different ontologies is that none of the taxonomies or thesauri have the same hierarchy for the same area as seen in Figure 5.3. Clearly, it is essential to develop a mechanism which can make the existing taxonomies suitable for an application domain, with more emphasis on the significance of context.

---

60 The research topics of the KA2 portal; http://ka2portal.aifb.uni-karlsruhe.de/ (2002).
Algorithm 8 describes the process of importing terms from taxonomies. For instance, we suppose that the term “ontologies” is included in the newly added document, then the system extracts all parents of the term up the hierarchy (i.e., predecessors) of a taxonomy (here “artificial intelligence” and “knowledge representation” from the hierarchy of Figure 5.3(c)). Next, the term “knowledge engineering” is triggered from the hierarchy of Figure 5.3(b) as the parent of the term “knowledge representation”. This process is repeated until no further terms are provided by the hierarchies. Then, the union set of the parent terms is suggested to the user in a list. The user is free to select any terms from the list to be added to the document.

```
Input: \( \Gamma \) - The set of keywords of a new adding document
Output: \( U \) - The union set of imported terms

Procedure importTerms(\( \Gamma \))
1 Begin
2 \( U \leftarrow \{ \} \)
3 For each taxonomy do
4 \( U_i \leftarrow \{ \} \)
5 For each element \( \gamma \) of \( \Gamma \) do
6 Find all parents of the term up the taxonomy hierarchy
7 \( U \leftarrow \) Union set of the parent terms
8 \( U_i \leftarrow U_i \cup U \)
9 For each element term \( t \) of \( U \) do
10 Repeat until the parent terms are empty
11 Find all parents of the term up the taxonomy hierarchy
12 \( U \leftarrow \) Union set of the parent terms
13 \( U_i \leftarrow U_i \cup U \)
14 End repeat
15 End for
16 End for
17 \( U \leftarrow U \cup U_i \setminus \Gamma \)
18 End for
19 Return \( U \)
20 End
```

**Algorithm 8.** The algorithm for importing terms from taxonomies.

### 5.3.3. Phase Three: Using co-occurred Terms in the Lattice

FCA is also used for knowledge acquisition (Wille 1992; Erdman 1998; Stumme 1998) and knowledge discovery in databases (Stumme et al. 1998; Hereth et al. 2000; Wille 2001) to discover concepts and rules related to objects and their attributes. This
approach is based on a strong idea of context with its use of parent child-relations between concepts in a graphically represented concept lattice. In common with most knowledge acquisition techniques, its power is in the way it presents relationships across the whole domain, and most FCA work attempts to display the whole lattice as noted in Chapter 4. The general principle is still to give the expert a view of the whole domain so that all relevant concepts will be included.

We have argued that experts more easily provide concepts that distinguish between cases (Compton and Jansen 1990). The expert’s attention is focused on relevant cases when the system misapplies a concept to a case. The expert is then asked to distinguish between this case and a case the system retrieves where the concept was appropriate. This is a more strongly situated view of knowledge acquisition with more emphasis on the significance of context. Motivated by considering specific objects and cases, we have implemented FCA in a similar way to both RDR and Repertory Grids. When a document is added keywords that co-occur with the keywords the user has assigned, are suggested by taking into account specific documents and keywords in the lattice. However, the user can navigate the whole lattice while adding a new document, as a view of the overall relationships between objects and attributes is also important.

The following describes how all co-occurring keywords are retrieved and then ranked for possible relevance before being presented to the annotator. To obtain keywords that are related with the added case in the lattice, the following definitions are used:

**Definition 4:** Let $C = (D, K, I)$ be a formal context and $\Gamma$ be a set of keywords ($\Gamma \subseteq K$). Then the set of documents associated with $\Gamma$ is defined to be $\Delta_{\Gamma} = \{d \in D \mid \exists k \in \Gamma \text{ such that } (d, k) \in I\}$.

$\Delta_{\Gamma}$ is introduced to get a set of documents, which have at least one keyword of $\Gamma$. If $\Gamma$ is a singleton (i.e., $\Gamma = \{\gamma\}$), then we will abbreviate $\Delta_{\gamma} = \Delta_{\Gamma} = \{d \in D \mid (d, \gamma) \in I\}$.

**Definition 5:** Let $C = (D, K, I)$ be a formal context. A function $f$ from $D$ to $2^K$ is defined as $f: D \rightarrow 2^K$ such that $f(d) = \{k \in K \mid (d, k) \in I\}$.
That is, $f(d)$ returns the set of keywords of $d$. Let the new document be $\delta (\in D)$ with the set of keywords $\Gamma$. We formulate the sub-formal context $C' = (D', K', I')$ with $D' = \Delta_c \cup \{\delta\}$ where $\Delta_c$ is as given in Definition 4 and $K' = \cup_{d \in D'} f(d)$ where $f$ is the function in Definition 5. In order to get keywords already associated with $\delta$, we first obtain a set of keywords which are associated with $\Delta_c$ as $f(\Delta_c) = \bigcup_{d \in \Delta_c} f(d)$ from the context $C'$. Now the set of co-occurred keywords is defined as $\Re = f(\Delta_c) - \Gamma$. Then, the function $W$ introduced below is used for each keyword $k$ of $\Re$ to compute the number of common keywords of $\Gamma$ with the keywords of all the documents that have the keyword $k$ from $C'$.

**Definition 6**: A function $W$ from $2^K \times \Re$ to the set of natural numbers $\mathbb{N}$ is defined as follows: $W: 2^K \times \Re \rightarrow \mathbb{N}$ such that $W(\Gamma, k) = \sum_{d \in \Delta_c} |f(d) \cap \Gamma|$ where $|X|$ is the cardinality of $X$.

Let us have a look this process in more detail. A user can annotate his/her document with a set of keywords by entering any terms or selecting given terms. The system displays all the keywords used by other annotators to facilitate sharing and reuse. After this initial assignment, the user can view the other terms that co-occur with the terms s/he has provided and can annotate the document with these further terms if desired. The terms are presented to the user ordered by their frequency and normalised for the number of terms at the node, and their “closeness” to the node to which the document is assigned by the user’s initial choice of terms in the conceptual hierarchy (i.e., weight).

More precisely, an ordered set of documents and a set of keywords which might be relevant to the new document are obtained. A sub-lattice $\mathcal{C}' (D', K', I')$ of the formal context $C'$ described above is then constructed. This step is divided into two stages. In the first stage, the relevant documents are obtained, ordered by their similarity with the new document. Given a new document $\delta$, we are interested in finding the set of documents that share some commonalties. We formulate a formal concept $\zeta = (\{\delta\}, f(\delta))$ with the newly added document $\delta$ and its keywords set $\Gamma$. Starting from the concept $\zeta$ we recursively go up to the direct superconcepts in the lattice to find the next level of relevant documents. This procedure is continued until the superconcept reaches the top node of the lattice.
Figure 5.4. A lattice $\mathcal{L}(D', K', I')$ of the formal context $C'$ from Figure 5.1.

For instance, suppose that there is a concept lattice as shown in Figure 5.1 and, a new document $\delta$ (6) is added together with its set of keywords $\Gamma$ {knowledge representation, ontology, knowledge management}. Then, we formulate the sub-context $C'(D', K', I')$ where $D' = \Delta \cup \{\delta\} = \{4, 5, 6\}$, $K' = \bigcup_{d \in D'} f(d) = \{\text{artificial intelligence, knowledge acquisition, knowledge representation, belief revision, ontology, knowledge management}\}$ and $I'$ is a binary relation between $D'$ and $K'$. The sub-lattice $\mathcal{L}(D', K', I')$ of the context $C'$ can be constructed as shown in Figure 5.4. The grey coloured box indicates the formal concept $\zeta$. From the lattice we can get document “5”, as it exists in the direct superconcept of $\zeta$ in the lattice, and as such is the most relevant to the document “6”. Next document “4” is obtained. Finally, an ordered set of documents {5, 4} relevant to document “6” in the lattice are obtained. The ordered documents are then viewed by the user along with the relevant features.

At the second stage, firstly, we elicit the terms that co-occur in the lattice with the terms the user has provided ($\mathcal{R}$). Secondly, a weight for each co-occurred term is calculated by Definition 6 ($W$). Next, the terms are ordered by their calculated weight and the ordered list is presented to the user with their weight. For example, let a new document $\delta$ be “6” and the set of keywords $\Gamma$ of $\delta$ be {knowledge representation, ontology, knowledge management}. Then, we get a set of documents associated with $\Gamma$, $\Delta = \{4, 5\}$ by Definition 4 from the sub-context $C'(D', K', I')$ as shown in Figure 5.4. Next,
the set of keywords which are associated with $\Delta r$ is obtained: that of $f(\Delta r) = \{\text{artificial intelligence, knowledge acquisition, knowledge representation, belief revision, ontology}\}$ by Definition 5. Following that, we obtain the set of co-occurred terms as $\mathcal{R} = f(\Delta r) - \Gamma = \{\text{artificial intelligence, knowledge acquisition, belief revision}\}$. Because the set of terms in $\mathcal{R}$ is a candidate for expanding the keywords already associated with $\delta$. Then, for each element of $\mathcal{R}$, a weight is calculated by Definition 6 as follows: $W(\Gamma, \text{artificial intelligence})=3, W(\Gamma, \text{knowledge acquisition})=2$ and $W(\Gamma, \text{belief revision})=1$.

Now, an ordered list is presented to the user with their weight. The user can select any keywords that are relevant. Through this process, the user can capture some relevant keywords while adding a new document. Of course, none of the mechanisms here are seen by the user, who sees only a ranked list of keywords. The user can also view the sub-lattice and the relevant documents for each of the co-occurred terms during this process.

5.3.4. Phase Four: Identifying related Documents

In the Ripple-Down Rules (RDR) method, an expert is only required to identify features that differentiate between a new case being added and the other stored cases already correctly handled. That is, the main technique of knowledge acquisition in RDR which is similar to the use of differences in Personal Construct Psychology (Gaines and Shaw 1990). A rule is only added to the system when a case has been given a wrong conclusion. Any cases that have prompted knowledge acquisition are stored along with the knowledge base. RDR does not allow the expert to add any rules which would result in any of these stored cases being given different conclusions from those stored unless the expert explicitly agrees to this. We import this RDR technique to help a user in finding appropriate keywords for documents.

When the assignment of keywords for a document is complete the document can be located at more than one node in the lattice. One node in particular is unique and has the largest intent among the nodes where the document is located. If there is another document already at the node, the user adding the new document is presented with the
previous document and asked to include keywords that distinguish the documents. The user can choose to leave the two documents together with the same keywords. Ultimately however, every document is unique and offers different resources to other documents and probably should be annotated to indicate the differences. The approach used is derived from RDR, but the location of the document is determined by FCA rather than the history of the development in RDR.

In the RDR approach, when a new rule is added, all stored cases that can reach the parent rule (cornerstone cases) are retrieved. Then the user is required to construct a rule which distinguishes between the new case and the cornerstone cases until it excludes all cornerstone cases. In this document retrieval system, a case which has the same set of keywords as the new document becomes the equivalent of a cornerstone case. If a cornerstone case exists, the system displays all the keywords used by other annotators. The user should select at least one different feature (keyword) from the deployed keywords or specify a new term to distinguish the cornerstone case(s) from the new case. This process is continued until the user is satisfied. A key difference from RDR is that RDR rules allow negations so that a child rule may be more general than its parent. This allows the historical dependencies of the rules to be maintained. Since the FCA lattice is constantly regenerated, it is reasonable for a distinguishing term to be added to a cornerstone case rather than the new case if preferred. However, this may also be referred to the owner of the original document or to a system manager.

5.3.5. Phase Five: Adding New Terms

Another mechanism of the annotation support tools to facilitate knowledge acquisition is triggered when a new term is entered for a new document; this term may also apply to other documents located at the parent nodes of the new node in the lattice. This problem could be left until the system fails to provide an appropriate document for a later search as in the RDR approach. However, in this proposed approach the system extracts those relevant documents at the direct parent nodes in the lattice and passes them to a knowledge engineer, who is able to examine whether the suggested documents should have the new keyword. The following definitions are used in formulating the relevant documents and their associated new terms:
**Definition 7:** Let $\mathcal{L} = \langle V, \leq \rangle$ be a lattice. Given a node $\theta \in V$, the set of direct parents of $\theta$ denoted $DP_\mathcal{L}(\theta)$ is defined as follows: $DP_\mathcal{L}(\theta) = \{ \alpha \in V \mid \theta < \alpha \text{ and there does not exist any } \beta \in V \text{ such that } \theta < \beta \land \beta < \alpha \}$.

**Definition 8:** Let $\mathcal{L}(C)$ be a concept lattice of the formal context $C = (D, K, I)$ and $\delta$ be the new document. For each document $d \in D$, the set of relevant keywords for $d$ with respect to $\delta$ denoted $Rel_\delta(d)$ is defined as follows:

$$Rel_\delta(d) = \bigcup_{X \in \delta} \{ f(\delta) \setminus (X,Y) \in DP_{\mathcal{L}(C)}((d),f(d))) \land Y \subseteq X \}$$

For instance, suppose that a new document $6$ ($\delta$) with the set of keywords $\Gamma$ \{knowledge representation, ontology, knowledge management\} is added into the lattice shown in Figure 5.1, then the lattice structure will be reformulated to cope with the new case. Figure 5.4 can be a part of a reconstructed lattice which has a new node $\zeta$ ($\{\delta\}, f(\delta)$) coloured grey. Now, for the documents located in the direct parent node of $\zeta$ (here document 5), we extract the relevant keywords with respect to $d$ by Definition 8: $Rel_\delta(d) = Rel_\delta(5) = \{\text{knowledge management}\}$. The system then passes this case to a knowledge engineer to determine whether or not document 5 should have the keyword “knowledge management”. Although it is possible to apply the term to higher parent nodes, for convenience only direct parents are considered.

**5.3.6. Phase Six: Logging Users’ Queries**

Another mechanism is activated when the system cannot find a node in the lattice with a user query. In this case, the system sends a log file to a knowledge engineer so s/he can decide if more appropriate keywords are required for documents through an interface supported by the system. If the knowledge engineer makes a decision, then the system automatically sends an e-mail to the author(s) (annotator of the document) with a hyperlink which can facilitate the refinement of the keywords of the document. All interactions between the system and users are also logged to find factors which may influence the search performance of the system.
5.4. Document Retrieval

Lattice-based retrieval is based on navigating the lattice structure of Formal Concept Analysis. In this approach, the lattice is used as a basic data structure either for indexing documents or for browsing. A node of the lattice consists of a concept with a pair \((X, Y)\) where \(X\) is the extent (a set of documents) and \(Y\) is the intent (a set of keywords) of the concept. The intents of each concept are used for the indexing terms of the browsing structure. Document retrieval in our approach followed this lattice-based model.

The central advantage of lattice browsing is that one can navigate down to a node by one path, and if a relevant document is not found one can go back up another path rather than simply starting again. When one navigates down a hierarchy, one tries to pick the best child at each step. If the right document is not found, it is difficult to know what to do next, because one has already made the best guesses possible at each decision point. However, with a lattice, the ability to go back up via another pathway to the node opens up new decisions, which one has not previously considered. The conventional hierarchical structure can be also embedded in this lattice structure.

Another strong feature of FCA for browsing is that the concept lattice holds the inheritance hierarchical relationship among the evolved attributes (keywords) in the lattice structure. The lattice also implies all minimal refinements and minimal enlargements for a query at an edge in the lattice (Godin et al. 1995). This means that following an edge upward (downward) corresponds to a minimal refinement (enlargement) for the query at the edge in the lattice, and vice versa. In other words, the intent (keywords) of each node can be considered as a conjunctive query, and the extent (documents) of the node is the search result with the query. Traversing edges upward from the query can deliver all minimal enlargements of the query in the lattice.

For example, let a user’s query be \(a \land c \land g \land h\) and the corresponding node (34, acgh) in Figure 5.5. The conjunctive intents \((a \land g \land h, a \land c)\) of the direct parents of the node (acgh) are all minimal enlargements of the query \((a \land c \land g \land h)\), and the conjunctive intents \((a \land c \land g \land h \land i, a \land b \land c \land g \land h)\) of the direct children nodes of the node are all
minimal refinements of the query. Therefore, the lattice structure can be used as a refinement tool for users’ Boolean queries in the evolved domain.

More importantly, the lattice with a set of documents and their keyword sets is scaled with an ontological structure for the attributes of the evolved domain (i.e., conceptual scaling). This allows a user not only to get more specific results, but also to search relevant documents by the interrelationship between the document keywords and the domain attributes. A more detailed explanation of the use of conceptual scaling will be presented in Section 5.5.

The user can also view the lattice using one of the imported taxonomies available in this case - the ACM, ASIS&T, Open Directory Project and UNSW taxonomy introduced in the previous section. The system recreates the lattice assuming that any document with a term from the imported taxonomy also has all the parent terms for that term. One can browse this lattice or alternatively one can navigate the lattice without any involvement of a taxonomy at any stage.

![Lattice Structure Diagram](image)

**Figure 5.5.** An example of a lattice structure.

*Numbers denote documents and alphabet characters indicate keywords. A node represents a concept with a pair \( (X, Y) \) where \( X \) is the extent and \( Y \) is the intent of the concept \( (X, Y) \).*
5.4.1. Browsing the Lattice Structure

A user can interact with the system starting from the root of the lattice exploring the relationships of the concepts from vertex to vertex of the lattice without any particular query being provided. We simplify the lattice display by showing only direct neighbour nodes using hyperlinks. The children and parents are hypertext links and a user navigates these links by clicking on a parent or child node. We can see how navigation is carried out, with a simple lattice structure as shown in Figure 5.5.

Suppose that the user’s query “a”, the system will display a set of documents of the concept “a” in a result space and will show the concepts “ag”, “ac”, “ab” and “ad” as more specialised nodes of “a” in a browsing space. Only direct neighbours of the node are displayed. If the concept “ac” was chosen, then the system will show its parent concept “a” and its child concepts “acgh”, “abc” and “acd”. Next, if we suppose that the concept “abe” was selected, then the parent concepts “ac” and “ab”, and the child concepts “abcgh” and “abcd” will be displayed. The user can again navigate up or down at this stage, or can move to the root of the lattice.

5.4.2. Entering a Boolean Query

The user can formulate a query by entering any text words in a conventional Boolean query interface or selecting terms from a list given by the system, and can navigate the lattice structure starting with a node covering the user’s query. A set of words can be separated by commas (,) assuming the AND Boolean operator. The query is normalised. In other words, firstly all stopwords are eliminated from the query. Secondly, the terms in the query are stemmed using the stemming classes. Following this, the system

---

62 Precisely speaking, “a” is the intent of the concept (12345678, a), but here we simply use the term concept, with only the intent of the concept. Note that the intents of concepts are used indexing terms for a lattice.

63 A knowledge engineer built a stopword list referring to a number of stopwords available on the Web.

64 The stemming classes have been downloaded from (http://ciir.cs.umass.edu/whatsnew/stemming.html, 2000). The entire 5.5-gigabyte TREC 1 - 5 collection was used to create the stemming classes by merging the Porter and K-Stem stemming algorithms which gave the overall best result on the TREC6 experiments. The purpose of the stemming is to deal mainly with the plural problem, rather than sophisticated morphological processing. We also added terms related to our test domains.
identifies a relevant node in the lattice with the normalised query and directly moves to the relevant portion. Note that when a lattice is formulated we also normalise the terms in each intent of the concepts in the lattice to match them with the normalised query.

To find a relevant node with the user’s query in the lattice we recall again that a formal concept is a pair \((X, Y)\) where \(X\) is the extent (a set of documents) and \(Y\) is the intent (a set of keywords) of the concept. The set of all formal concepts in the concept lattice of a context \(C\) is denoted as \(\mathcal{B}(C)\) and the set of all intents of \(\mathcal{B}(C)\) as \(\text{int}(\mathcal{B}(C))\). Now, let a user’s query be \(Q\), if a concept \(c \in \mathcal{B}(C)\) satisfies the following conditions, then \(c\) is the relevant portion of the query \(Q\) in the lattice of \(C\).

(i) \(Q \subseteq \text{int}(c)\)

(ii) For each set of keywords (intent) \(\alpha \in \text{int}(\mathcal{B}(C))\) if \(Q \subseteq \alpha\), then \(\text{int}(c) \subseteq \alpha\)

For instance, if we take the lattice shown in Figure 5.5 and a user’s query “\(a \land c\)”, then the node \((34678, ac)\) will be the starting point of the navigation with the query. The system will display a set of documents \((3,4,6,7,8)\) of the concept “\(ac\)” in a result space, and its parent concept “\(a\)” and its child concepts “\(acgh\)”, “\(abc\)” and “\(acd\)” in a navigation space. With a query “\(a \land b \land d\)”, the node \((56, abdf)\) will be the starting point of the navigation.

A relevant portion may not exist in the lattice with a given query \(Q\). In this case, documents are retrieved which contain the query anywhere in the contents of the documents (text words search), and the system formulates a sub-lattice using the result documents and their keywords. Navigation can be done on this sub-lattice.

To provide more flexible retrieval options we also display the keywords which subsume the user’s query if such keywords exist. For instance, suppose that a user’s query is “compiler”, then the system will display the search results of the query, and also display keywords that subsume the query such as “compiler construction”, “complier techniques”, “dynamic compliers” and “incremental compilers” \(\supseteq\) compiler. These are shown as a list and are hypertext links to the appropriate part of the lattice. The
standard retrieval mechanism on the concept lattice can be considered as phrase searching\(^{65}\) combined with the AND Boolean operator. For example, let us take a node \((\{1, 2, 5\}, \{\text{artificial intelligence, knowledge acquisition}\})\) from Figure 5.1. Then, it can be regarded that the documents 1, 2 and 5 are the search results for the query which consists of two phrases “artificial intelligence” and “knowledge acquisition” with a conjunction of two phrases (i.e., “artificial intelligence” AND “knowledge acquisition”). Thus, the deployment of the subsuming keywords of a user’s query will be useful when the user’s query is a part of the phrased keywords.

### 5.5. Conceptual Scaling

Conceptual scaling has been introduced in order to deal with many-valued attributes (Ganter and Wille 1989; 1999). According to the basic theory of conceptual scales of FCA, each attribute, or a combination of more than one attribute of the many-valued context, can be transformed into a one-valued context. The derived one-valued context is called a conceptual scale. Then, if one is interested in analysing the interrelationship between attributes, s/he can choose and combine the conceptual scales which contain the required attributes. This process is called conceptual scaling.

More fundamentally, conceptual scaling is to deal with many-valued Boolean attributes which hold multiple inheritance within a one-valued context of FCA. The essence of conceptual scaling is to impose on this a single inheritance hierarchy or equivalently some of the Boolean attributes are reorganised as being mutually exclusive values of some unnamed attributes. Either way there is recognition that a group of Boolean attributes are mutually exclusive. In conceptual scaling, one selects one of the mutually exclusive attributes from a set and a sub-lattice containing these values is shown. A number of attributes selection can be made at the same time to give the sub-lattice. Existing attributes can be used as the parent of group of mutually exclusive attributes or new names for the grouping can be created.

\(^{65}\) Phrase searching is a feature which allows a user to find documents containing certain phrases. When phrase searching is used, only documents which contain the phrase are retrieved.
There are two ways in which we use conceptual scales in the proposed system. Firstly, a user or a system manager can group a set of keywords used for the annotation of documents. The groupings are then used for conceptual scaling. Secondly, other ontological information can also be used where readily available (e.g., person, academic position, research group and so on). These correspond to the type of more structured ontological information used in the system such as KA² (http://ka2portal.aifb.uni-karlsruhe.de/). The key point of the proposed approach is flexible evolving ontological information but there is no problem with using more fixed information if available. We have included such information for interest and completeness in conceptual scaling. These conceptual scales allow a user to get more specific results and to reduce the complexity of the visualisation of the browsing structure as well as to search relevant documents by the interrelationship between the domain attributes and the keywords of documents.

An intended purpose of conceptual scaling is to support a hybrid browsing approach by connecting an outer structure with keyword sets of documents (taxonomies) and a inner nested structure with ontological attributes (ontological structure). The ideal would be to support both approaches simultaneously because the organisation of background knowledge, not only with the vocabularies in taxonomies but also with ontological structures in the form of properties, would be useful for navigating information.

It should be noted that, in referring to the term “ontology” here, we are neither dealing with a formal ontology which uses relations, constraints, and axioms, nor providing automated reasoning based on implied inter-ontology relationships. Rather our aim is a browsing mechanism suitable for specialised domains.

Conceptual scaling will be explained using examples for the domain of research interests. In the domain of research interests, $D$ is the set of home pages and $K$ is the set of research topics for a context ($D, K, I$). However, the word *documents* and *keywords* are also used interchangeably to denote *home pages* (or simply *pages*) and *research topics* (or simply *topics*), respectively.
5.5.1. Conceptual Scaling for a Many-valued Context

A many-valued context is defined as a formal context $C = (D, M, W, I)$ where $D$ is a set of documents, $M$ a set of attributes, $W$ a set of attribute values. $I$ is a ternary relation between $D$, $M$ and $W$ which indicates that an document $d$ has the attribute value $w$ for the attribute $m$. We formulate a concept lattice with a set of documents and their keywords as shown in Figure 5.1. This lattice structure is the main browsing space, but is also an outer structure. Other attributes in a many-valued context are then scaled into a nested structure of the outer structure at retrieval time.

Table 5.2 is an example of a many-valued context in the domain of research interests. The attributes in the many-valued context can be represented in a partially ordered hierarchy as shown in Figure 5.6. The attribute “position” in Table 5.2 is located as a subset of the attribute “person” in the hierarchy. To explain this in a more formal way, the following definition is provided.

**Definition 9**: Let $S_p$ be a super-attribute and $S_c$ be a sub-attribute. There is a binary relation $\mathcal{R}$ called the “has-value” relation on $S_p$ and $S_c$ such that $(p, c) \in \mathcal{R}$ where $p \in S_p$ and $c \in S_c$ if and only if $c$ is a sub-attribute value of $p$.

For example, the has-value relation $\mathcal{R}$ on the attributes “person” and “position” is: $\mathcal{R} = \{(\text{academic staff, professor}), (\text{academic staff, associate professor}), \ldots, (\text{research staff, research assistant}), \ldots, (\text{research student, Ph.D. student}), (\text{research student, ME student})\}$ from Figure 5.6. This hierarchy of the many-valued context with the relation $\mathcal{R}$ is scaled into a nested structure using pop-up and pull-down menus.

<table>
<thead>
<tr>
<th>Researcher</th>
<th>Research group</th>
<th>Sub-group of AI</th>
<th>Person</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Researcher1</td>
<td>Artificial intelligence</td>
<td>Knowledge Acquisition</td>
<td>Academic staff</td>
<td>Professor</td>
</tr>
<tr>
<td>Researcher2</td>
<td>Computer systems</td>
<td>.</td>
<td>Research staff</td>
<td>Research associate</td>
</tr>
<tr>
<td>Researcher3</td>
<td>Networks</td>
<td>.</td>
<td>Academic staff</td>
<td>Associate professor</td>
</tr>
<tr>
<td>Researcher4</td>
<td>Databases</td>
<td>.</td>
<td>Academic staff</td>
<td>Senior lecturer</td>
</tr>
<tr>
<td>Researcher5</td>
<td>Software engineering</td>
<td>.</td>
<td>Research students</td>
<td>Ph.D. student</td>
</tr>
</tbody>
</table>

*Researchers can be the objects of the context as they are the instances of the home pages.*
Figure 5.6. Partially ordered multi-valued attributes for the domain of research interests.

Figure 5.7 shows examples of inner browsing structures corresponding to concepts of the outer lattice. A nested structure is constructed dynamically from the extent (home pages) of a corresponding concept of the outer lattice incorporating the ontological hierarchy. When a user assigns a set of topics for their page, the page is also automatically annotated with the values of the attributes in the many-valued context. A default home page for individual researchers is provided at the School Web site as well as every researcher has a login account at the School. We make to use this login account when a user annotates their home page. This provides the default home page address of the user. The page is an HTML file in a standard format including the basic information of the researcher such as their first name, last name, e-mail address, position and others. The system parses the HTML file and extracts the values for the pre-defined attributes. From the attributes and their extracted values, we formulate a nested structure for a concept of the lattice at retrieval time.
Note that the attributes which do not exist in the default home page can be used for conceptual scaling. The user will need to be presented to annotate the values of those attributes when they assigns a set of keyword for their document. Here we recognise that there will be significant issues relating to the annotation bottleneck as an ontological approach presented in Chapter 2. However, in the proposed approach, the user will not require an understanding of the notions of ontology such as relations, constraints and axioms as in an ontological approach. The user will be given a simple interface to click selection of values or a series of text boxes to be filled.

Figure 5.7. Examples of nested structures corresponding to concepts.

This shows nested structures corresponding to the concept “artificial intelligence” and “artificial intelligence, machine learning” of the outer structure which is constructed with a set of home pages and their topics. Numbers in the lattice and in brackets indicate the number of pages corresponding to the concept of the lattice and the attribute value, respectively. Here, the nested structure is presented in a hierarchy deploying all embedded inner structures. But the structure is implemented using pop-up and pull-down menus as shown in Figure 5.8.
A user can navigate recursively among the nested attributes observing the interrelationship between the attributes and the outer structure. By selecting one of the nested items, the user can moderate the cardinality of the display. Again, the structure with the most obvious attributes can be partly equivalent to the ontological structure of the domain and consequently is considered as an ontological browser which is integrated into the lattice structure with the keywords set.

Figure 5.8 shows an example of pop-up and pull-down menus for the nested structure of the concept “artificial intelligence” in Figure 5.7. The menu of ① appears when a user clicks on the concept “artificial intelligence”. Each item of menu ① is equivalent to a scale in the many-valued context. Suppose that the user selects the attribute Person in menu ①, the system then will display a sub-menu of the attribute as shown in menu ④. Note that the menu items of ④ are the values of the attribute Person in the many-valued context in Table 5.2. If we assume that the menu item “academic staff” is selected, then the menu of ⑤ will appear. The menu items of ⑤ are the values of the attribute Position that are in a binary relation ℜ on the “academic staff” by Definition 9. The search results will be changed according to the selection of an item of a menu.

Figure 5.8. An example of pop-up and pull-down menus for the nested structure of a concept.

5.5.2. Conceptual Scaling for a One-valued Context

Conceptual scaling is also applied to group relevant values in the keyword sets used for the annotation of documents. The groupings are determined as required, and their scales
are derived on the fly when a user’s query is associated with the groupings. This means that the relevant group name(s) is included into the nested structure dynamically at run time. Table 5.3 shows examples of groupings for scales in the one-valued context for the attribute *keyword*.

**Table 5.3.** Examples of groupings for scales in the one-valued context.

<table>
<thead>
<tr>
<th>Grouping names (Generic terms)</th>
<th>The members of the grouping names</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDR</td>
<td>FRDR, MCRDR, NRDR, SCRDR</td>
</tr>
<tr>
<td>Sisyphus</td>
<td>Sisyphus-I, Sisyphus-II, Sisyphus-III, Sisyphus-IV, Sisyphus-V</td>
</tr>
<tr>
<td>Knowledge acquisition</td>
<td>Knowledge acquisition methodologies, Knowledge acquisition tools, Incremental knowledge acquisition, Automatic knowledge acquisition, Web based knowledge acquisition, …</td>
</tr>
<tr>
<td>Computer programming</td>
<td>Concurrent programming, Functional programming, Logic programming, Object oriented programming, …</td>
</tr>
<tr>
<td>Programming languages</td>
<td>Concurrent languages, Knowledge representaion languages, Logic languages, Object oriented languages, …</td>
</tr>
<tr>
<td>Databases</td>
<td>Deductive databases, Distributed databases, Mobile databases, Multimedia databases, Object oriented databases, Relational databases, Spatial databases, Semistructural databases</td>
</tr>
<tr>
<td>Natural language</td>
<td>Natural language processing, Natural language understanding</td>
</tr>
<tr>
<td>Web</td>
<td>Web applications, Web Searching, Web services, Web operating systems, …</td>
</tr>
<tr>
<td>XML</td>
<td>XML applications, XML tools, …</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Applied to a one-valued context, the following definition is provided:

**Definition 10:** Let a formal context \( C = (D, K, I) \) be given. A set \( G \subseteq K \) is a set of grouping names (generic terms) of \( C \) if and only if for each keyword \( k \in K \), either \( k \in G \) or there exists some generic term \( \kappa \in G \) such that \( k \) is a sub-term of \( \kappa \). We define \( S = K \setminus G \) and a relation \( gen \subseteq G \times S \) such that \((g, s) \in gen\) if and only if \( s \) is a sub-term of \( g \).
Then, when a user’s query is \(qry \in G\), a sub-formal context \(C' = (D', K', I')\) of \((D, K, I)\) is formulated where \(K' = \{k \in K \mid k = qry \text{ or } (qry, k) \in \text{gen}\}\), \(D' = \{d \in D \mid \exists k \in K' \text{ and } dk\}\) and \(I' = \{(d, k) \in D' \times K' \mid (d, k) \in I\} \cup \{(d, qry) \mid d \in D' \text{ and } qry \in K' \cap G\}\). For instance, suppose that there are groupings as shown in Table 5.3 and a user’s query “databases”. The query databases \(\in G\) so that a sub-context \(C'\) is constructed to include a scale of the grouping name databases and build a lattice of \(C'\). The user can then navigate this lattice of \(C'\).

Figure 5.9 shows an example of a scale with the grouping name “databases”. The grouping name is embedded into an item of the nested structure along with other scales from the many-valued context in the previous section. There are 10 documents with the concept “Databases” in the lattice, and the node (Databases, 10) embeds the scales as shown in menu 1. The scale “Databases” was derived from the groupings in the one-valued context, while other scales (items) were derived from the many-valued context (i.e., domain attributes). A user can read that there is one document related to “deductive databases”, and two documents with “multimedia databases” etc. By selecting an item of sub-menu 2, the user can moderate the retrieved documents which are only associated with the selected sub-term.

**Figure 5.9.** A conceptual scale for the grouping name “databases”.
The reason for formulating a sub-formal context $C'$ is that the lattice used for the outer structure (with a set of documents and their keywords) does not include a node which subsumes all documents related to the set of sub-terms of a grouping name, because the documents associated with the sub-terms of a grouping may or may not be related to the grouping name (i.e., generic term). Thus, we formulate the context $C'$ to have a relation between the grouping name and the documents which are associated with at least one of the sub-terms of the grouping. A lattice is then derived from the context $C'$.

As a consequence, a node, which contains all documents associated with the members of the evolved grouping name, is contained in the sub-lattice. In this approach, we may lose the advantage of lattice-based browsing that allows a user to navigate the whole lattice freely exploring the domain knowledge. Because the space of navigation is limited within the sub-lattice, the system even supports a link to start navigation in the whole lattice at any stage. The following method therefore can be used as an alternative.

A knowledge engineer/user can set up or change the groupings using a supported tool (i.e., ontology editor) whenever it is required. When a grouping name with a set of sub-terms is added, the system gets the set of documents that are associated with at least one of the sub-terms of the grouping name. Then, the context $C$ is refined to have a binary relation between the grouping term and the documents related to the sub-terms of the grouping term. Next, the lattice of $C$ is reformulated when any change in $C$ is made. If a grouping name is changed, it is replaced with the changed one in the context $C$ and its lattice.

In the case of removal of a grouping in the hierarchy, no change is made in the context $C$. With this mechanism, the outer lattice can always embed a node which can assemble all documents associated with the sub-terms of a grouping. That is, the groupings play the role of intermediate nodes in the lattice to scale the relevant values. Groupings can be formed with more than one level of hierarchy. This means that a sub-term of a grouping can be a grouping of other sub-terms.
5.6. Chapter Summary

We presented an incremental domain-specific document management and retrieval system based on lattice-based browsing of Formal Concept Analysis and outlined the functionality of the system that we proposed. We focused on a Web document management system for small communities in specialised domains based on free annotation of documents by users. Another main focus was an emphasis on incremental development and evolution of the system. A number of knowledge acquisition techniques were developed to suggest possible annotations, including suggesting terms from external ontologies. Lattice-based browsing was incrementally constructed as the users annotate their documents and used as the basic structure for retrieval.

Document retrieval for end-users is based on browsing this lattice structure. Users can interact with the system starting from the root of the lattice and exploring the relationships of the concepts from vertex to vertex of the lattice without any particular query being provided. The lattice display was simplified by showing only direct neighbour lattice nodes using hyperlinks for a Web-based system. The user can also formulate a query by entering any text words in a conventional Boolean query interface or selecting terms from a list supported by the system, and can navigate the lattice structure starting with a node covering the user’s query.

More importantly, the lattice was combined with a hierarchical ontological structure to allow a nested structure at retrieval time dynamically, referred to as conceptual scaling. In essence the conceptual scales give a view of a lattice formed from objects that have specified attribute value pairs. Conceptual scaling was also used in a one-valued context (i.e., the attribute keyword) to group relevant values in the keywords set. The groupings are determined as required, and their scales are derived on the fly when a user’s query is associated with the groupings.

The user can also view the lattice using one of the imported taxonomies available. This recreated the lattice assuming that any object with an attribute from the imported taxonomy also has all the parent terms for that term.
To demonstrate the value of the proposed approach, we conducted experiments in the domain of research topics in the School of Computer Science and Engineering (CSE), University of New South Wales (UNSW). We also set up a system that allows users to annotate papers from the on-line Banff Knowledge Acquisition Proceedings. The systems are presented in the next chapter.
Chapter 6
Implementation

Prototypes have been implemented on the World Wide Web to demonstrate and evaluate the proposed approach. The first system is intended to assist in finding research topics and researchers in the School of Computer Science and Engineering (CSE), University of New South Wales (UNSW)\(^ {66}\). The goal was a system to assist prospective students and potential collaborators in finding research relevant to their interests. There are around 150 research staff and students in the School who generally have home pages indicating their research topics. The system allows staff and students to freely annotate their home pages so that they can be found within an evolving lattice of research topics. The second implementation is a system\(^ {67}\) that gives access to the on-line Banff Knowledge Acquisition Proceedings with around 200 publications in recent years\(^ {68}\).

The system will be described mainly with reference to the domain of research interests. In the domain of research interests, a document corresponds to a home page and a set of keywords is a set of research topics.

Section 6.1 gives an overview of the system we propose. Section 6.2 outlines the basic environment of the system. The implementation with the domain of research interests is described in Section 6.3. We present how documents associated with the annotation mechanisms can be managed by users themselves, and how the annotated documents can be searched using both browsing and Boolean queries. The system for the domain of the Banff Knowledge Acquisition Proceedings is presented briefly in Section 6.3.2.

\(^{66}\) URLs of the system: \url{http://www.cse.unsw.edu.au/search.html} and \url{http://www.cse.unsw.edu.au/school/research/index.html} pointing to the \url{http://pokey.cse.unsw.edu.au/servlets/RI}.

\(^{67}\) \url{http://pokey.cse.unsw.edu.au/servlets/Search}.

\(^{68}\) KAW96, KAW98 and KAW99 (\url{http://ksi.cpsc.ucalgary.ca/KAW/}, 2000).
6.1. Overview of the System

Figure 6.1 shows the architecture of the system we developed for a domain-specific document management and retrieval system. The system has two main functions - a “document management engine” and a “document retrieval engine”.

The “document management engine” builds and maintains knowledge bases for documents and a concept lattice for browsing. Users themselves annotate their own documents with a set of keywords using knowledge acquisition mechanisms (i.e., annotation support tools) that aim to capture the concepts which are missed or unknown when the keywords are first assigned for a document.

When a user annotates their document, they can select keywords already used in the system which have been added by others or enter further textwords which in turn will be available to future users. In other words, the user is provided with a list of keywords already available. After an initial selection, the user can view other terms that are imported from other taxonomies. The system extracts all parents of the term up the hierarchies of taxonomies which are related to the initial selection and presents them to the user. The system also indicates keywords that have been used together with the keywords already selected for other documents in the lattice structure. Through these and further knowledge acquisition steps, the initial keywords can be refined.

Figure 6.1. Architecture of the system.
Then, the case (a document with a set of keywords) is added into the system, triggering the update of the concept lattice which is used as a basic data structure for indexing documents and browsing in our approach. This concept lattice is incrementally and automatically reformulated whenever a new case is added or existing cases are changed. Figure 6.2(a) shows an example of a lattice.

The second main function is a “document retrieval engine” for finding documents constructed in the concept lattice. The user can browse the lattice structure to find information. The user can also formulate a query by entering any textwords in a conventional information retrieval fashion or by selecting a keyword from those that had been used for annotating the documents. If a keyword has been selected or textwords identify some keywords, the system identifies the appropriate node and displays it together with its direct neighbours. The user can start navigation from this node.

If the system does not include a node with the given keywords, it displays a sub-lattice which covers documents that contain the textwords anywhere in the document. The user can navigate this sub-lattice, and also transfer to the same node in the overall lattice. If the textwords entered do not correspond to a node, the system also sends a log file to an expert so they can decide if more appropriate keywords are required for the documents.

A knowledge engineer can define the attributes of the evolved domain with a partially ordered hierarchy among the attributes. This requires a prior domain ontology in the same way as (KA)² and is included in our system only for completeness. We suggest that it will be used only for the most obvious attributes rather than for implementing a fully developed ontology. When a user annotates his/her document, the system then automatically extracts the values of the attributes defined from the content of the annotated document. The attributes and their values are accessed via nested browsing as shown in Figure 6.2(c). The concept lattice with a set of documents and their keyword sets becomes the outer structure as shown Figure 6.2(b) and serves as the main navigation space. The structure of Figure 6.2(c) is nested in a corresponding concept of the outer lattice on the fly. That is, nested browsing is constructed dynamically at run
Figure 6.2. An example of a browsing structure.

(a) Lattice structure. (b) Indexing of the lattice. (c) Nested structure. (d) A home page (URL).

Time from documents belonging to a corresponding concept of the outer lattice and based on the structure of the attributes defined. This provides conceptual scaling between the domain attributes and the search results with the keywords set. It allows the user to obtain more specific search results, reducing the complexity of the navigation space. For instance, the user can read that there is a researcher whose research topic is “Artificial intelligence” and her position is “Professor”.

Once again, the system can be explored at: http://pokey.cse.unsw.edu.au/servlets/RI and http://pokey.cse.unsw.edu.au/servlets/Search. The following section outlines the basic environment of the system.

---

69 Numbers in parentheses indicate a number of documents corresponding to the attribute value.

70 A document is connected to an HTML page.
6.2. Basic Environment of the System

The system was developed with Java, JavaScript and Java Servlets (Java CGI: Common Gateway Interface). The internal structure of the system is composed of a Web server, and an interface environment based on a client/server architecture. The server is written as a CGI library (Java Servlets) and Java. The interface is based on HTML supported by a Web browser such as Netscape 4.0 and Explorer 5.0 or higher.

Security

Anyone can access and browse the lattice to find information within the system. However, for annotations, only staff and research students of the School of Computer Science and Engineering, UNSW can annotate pages for research topics as the only documents the system provides access to, are the home pages of these staff and students. We use the local Unix account at the School to authenticate users for the annotation. This also provides a default home page address for the users. This security system is specific to this application and different approaches will be required for other applications.

Annotation Mechanism

Only annotations and the URLs of the pages are stored on our local server. Further development of the project would probably look at using encoded annotations from within the document as well as having annotations stored on the server. It requires a new version of the document which is marked up with annotations from the server.

Browsing Structure Generation

The system has an automatic document clustering feature which creates its clusters using terms taken from the annotated documents. We construct a conceptual lattice browsing structure which relates documents and clusters (keywords) as well as showing relationships among documents and among keywords. The system updates the browsing structure (concept lattice) whenever a new document is added with a set of keywords or when the keywords of existing documents are refined. This is essential if users are to get immediate feedback on the clusters that emerge from changes in annotation.
**User Interface for Browsing**

The system has a Web interface, and the lattice for browsing is simplified by showing only direct neighbours in the lattice using hyperlink techniques. The children and parents are hypertext links and the user navigates by clicking on parent and children nodes. Hypertext links to documents associated with the current node are also shown along with a brief summary of the page.

**Knowledge Engineering Support**

Although the system supports annotation by users without intervention of a knowledge engineer, the system also supports the notion of a domain manager who can make some behind the scene changes to improve the functionality of the system. But the role of the knowledge engineer can be reduced and/or replaced by the user.

### 6.3. Presentation of the System

Section 6.3.1 will present the system with reference to the domain of research interests in detail. Following that, the system for the domain of the Banff Knowledge Acquisition Proceedings will be described briefly in Section 6.3.2.

### 6.3.1. Domain of Research Interests in a Computer Science School

#### 6.3.1.1. Document Annotation

A researcher can annotate their own home page with a set of research topics by selecting among the topics already used or by freely specifying new topics through given interfaces. When the researcher logs onto the system using the local Unix account, the system authenticates the user. If the user is identified, the system extracts the basic information of the user (e.g., name, phone number, fax number, e-mail address and homepage address) from his/her default home page address and displays the annotation screen as shown in Figure 6.3. Note that a default home page for individual researchers is provided at the School Web site in an HTML file. The system parses the HTML file of the user and extracts the values for the pre-defined attributes.
Figure 6.3. An example for the annotation of a home page.

When a researcher logs on to the system, the above screen will be displayed for the annotation of the page. The rest of the screen displays the further topics used by other researchers plus those contained in the imported taxonomies.

Topics are initially selected by clicking the checkbox in front of each term, and/or entering any new topics. To assist in finding relevant topics from those already used, the researcher can select from the topics used by other researchers with whom they may share interests. This can be done via the link from other UNSW researchers in the above screen. Some researchers would like to examine the annotated research topics of their collaborators. The annotator (researcher) can see a list of topics based on each of the selected researchers as shown in Figure 6.4.
First the annotator needs to select researchers to view their research topics through a given interface. The system then will display topics based on the selected researchers as above. The annotator can choose topics by clicking the checkbox of each term s/he would like to assign as topics. After the annotator has selected some terms, s/he is then presented with a display of terms that are imported from other taxonomies and that co-occur with the selected terms in the lattice as shown in Figure 6.5. The purpose is to prompt them to consider groupings of terms used by others that may be related. Some topics may be “made up” in collaboration with other researchers and/or research groups. The researcher can annotate the page with these further terms if desired.
Figure 6.5. An example of displaying possible relevant topics for the page being annotated.

In the above screen, the hyperlink Hendra Suryanto (the researcher being annotated) is connected to the annotator’s home page. The link relevant pages shows documents ordered by a similarity with the research topics of the annotator and the link sub-lattice shows a sub lattice of these pages. The taxonomy links (UNSW, ACM, ASIS) take one to the hierarchy of each taxonomy and the research topic links listed take one to documents (i.e., researcher pages with these topics). The numbers in the parenthesis indicate the relevancy weight of the topic to the annotator’s initial choice of topics.
The terms suggested from the external taxonomies are extracted from the ACM computing classification taxonomy and ASIS&T thesaurus for information science. They are also extracted from the UNSW taxonomy which has been developed using the hierarchical clusters of the Open Directory Project, the KA\textsuperscript{2} community Web site, and the research areas at the School of Computer Science and Engineering, UNSW. When a term from the initial assignment of topics occurs in one of these hierarchies, the system shows all the parents of this term up the hierarchy (i.e., predecessors). However, the results from the various hierarchies are merged into a single list.

The terms “Learning” and “Knowledge Engineering” in Figure 6.5 are the parents of terms in the taxonomical hierarchies of the topics that the annotator had initially assigned. Any of these terms can be selected by the annotator and added to the document. Note that in an inheritance sequence, the user is free to pick any or none of the parent term up the hierarchy. For example, a general parent may be selected, but the immediate parent may be omitted. Relationships between terms evolve dynamically and are determined by Formal Concept Analysis, rather than being constructed from the pre-existing hierarchical clusters for general situations (purposes).

By taking into account specific pages (documents) and topics in the lattice, other terms are suggested that co-occur with the topics the annotator has assigned. These terms are presented to the annotator ordered by their weight which is normalised for the number of terms at the node, and their “closeness” to the node to which the page is assigned by the annotator’s initial choice of terms. Again the annotator simply clicks the check box located in front of each term to select it.

At this stage the annotator can view the set of documents ordered by a similarity measure\textsuperscript{71} in the lattice with the current page as shown in Figure 6.6. As well, the annotator can observe a sub-lattice constructed of relevant documents by clicking a hyperlink on this screen. The annotator can also view the pages listed alphabetically for each of the related topics as well as the existing lattice structure. Through these processes, the annotator may find other relevant topics s/he has missed.

\textsuperscript{71} See Section 5.3.3 for details.
Figure 6.6. An example of relevant pages with the page being annotated.

The hyperlinks are connected to the researchers’ home pages.

After these procedures, the page (document) can be located at more than one node in a lattice. One node in particular is unique and has the largest intent among the nodes where the page is located. If there is another page already at the node, the annotator is presented with the previous page and is given the opportunity to include topics that distinguish themselves with the previous page. Figure 6.7 shows an example of this. A further page may be prompted by the newly added topics. So that this process will be continued, until there is no further page that has the same keyword set (topics) as those of their page. The annotator can choose to leave the two pages together with the same topic. Ultimately however, every home page is unique and offers different resources to other pages and probably should be annotated to indicate the differences.
Figure 6.7. An example of identifying related pages.

This shows a stored case (in the above screen called a cornerstone case) that matches the current case being added. Topic(s) can be added to differentiate two cases by adding any new term and/or selecting the terms in the check boxes as before.

When the above stage is complete, the concept lattice is automatically rebuilt and the page (document) is located at a node of the lattice. The annotator can immediately view the concept lattice that incorporates his/her page and further decide whether the set of topics s/he assigned for the page are appropriate. The navigation process itself can be a learning process for the annotator to capture and discover domain knowledge, and can influence keyword choices.
6.3.1.2. System Maintenance by a Knowledge Engineer

Even though a user can annotate his/her page without the intervention of a knowledge engineer (or manager), the system supports the use of a knowledge manager who can make some changes to improve the functionality of the system.

The knowledge manager receives reports of all new terms entered as it is possible that any pages located at parent nodes to the node with the new term should perhaps also be annotated with this term. In other words, when a new term is entered for a new document; this term may also appropriately apply to other documents already in the system. In this case the system extracts those relevant documents (pages) at the direct parent nodes of the new node in the lattice and passes them with the new term(s) to the manager. The manager decides whether it is appropriate to contact the owners of these pages to see if they want to use the new annotation. Figure 6.8 shows an example of this occasion.

![Image of system maintenance interface](image.png)

**Figure 6.8.** An example of adding new terms.
For example, by adding a new case (the researcher “Akara Prayote” and his research interests), the topic “network fault diagnosis” may be relevant to the researcher “Paul Compton” and “Abdus Khan” located at the parent node of the new node. If the manager selects the suggested topic and clicks on the “Save” button, the system will create e-mail and send it to the respondent researcher to facilitate the assignment of the suggested topic if desired.

Another mechanism is activated when the system cannot find a node in the lattice for a user’s query. If a user searches with a term that is not a keyword used for the annotations, a textword search is carried out. In this case a report is sent to the knowledge manager as this may suggest that a new keyword needs to be added to the system. If the manager makes a decision for the case(s), the system creates an e-mail automatically and sends it to the author (annotator of the document). It includes a hyperlink which can facilitate the refinement of the keywords of a document if desired.

As the system evolves, new terms are added. As a consequence, there is a necessity to handle synonyms, abbreviations or to group relevant terms together for extending a user’s query. The knowledge manager has access to a tool (i.e, ontology editor), which allow him/her to identify abbreviations, synonyms or groupings. A fairly simple and standard graphic editor is available for this task.

The screen in Figure 6.9 shows how relevant terms are grouped and edited. The manager can set up partial hierarchies so that related terms can be grouped under a common name. For example, the terms “Deductive Databases”, “Distributed Databases”, “Mobile Databases”, “Object Oriented Databases” and “Relational Databases” can be grouped with the name “Databases”. Then, when a user’s query is relevant to the term “Databases”, all the documents that include the terms belonging to the group “Databases” are retrieved and a nested structure which represents the group hierarchy is also supported. Note that this is different from the search feature which shows all keywords that contain a given sub-string (see Figure 5.9 in Chapter 5 and Figure 6.11). The knowledge manager can also edit synonyms and abbreviations. If a user’s query uses one of these synonyms (or abbreviations), the system extends the query based on the relevant synonym.
This shows a snapshot of editing a group name “Databases”. The left-hand side of the screen is the browser for groupings. The knowledge manager can also edit synonyms and abbreviations using the link Edit Synonym and Edit Abbreviation, respectively.

6.3.1.3. Document Retrieval and Browsing

The main search mechanism is based on browsing a concept lattice of FCA. Browsing is based on showing a Web page with the hyperlinks. A user can interact with the system starting from the vertex of the lattice and exploring the relationships of the concepts (topics) without any particular query being provided. Figure 6.10 shows the top-level concepts of the lattice.

The browsing structure here can be different with the structure of the on-line system as the system evolves. As well, with limited screen size, some branches of the lattice are omitted in the example Figures.
Figure 6.10. A snapshot of browsing the top-level concepts.

A text box for entering topics is shown. A complete list of concepts is also shown. The concepts (topics) are hyperlinks to that concept node in the lattice. The numbers in brackets indicate the number of researchers at each node.

Navigating the lattice, users can select terms from supported topics or enter terms into a text box. This means that the user can specify a query by entering any textwords in a conventional information retrieval fashion or by selecting a term among those already used for annotating documents. A set of words can be entered separated by commas ("," ) assuming the AND Boolean operator. Stopwords are first eliminated and the remaining query is stemmed using the stemming classes. If the entered term is a keyword, the system identifies the most relevant portion in the lattice for the query and moves to this node - displaying only the direct neighbours of the node. Figure 6.11 shows the search result when the user selected the hyperlink “Artificial Intelligence (39)” from Figure 6.10 or entered a query with “artificial intelligence”.

124
Figure 6.11. An example of a browsing structure.

Figure 6.11 shows the search result with the term “artificial intelligence”. The URLs for these researchers can be accessed via the folders on the left. The researchers for the current node are also listed at the bottom of the screen (shown partly). The “Nested” button gives a Conceptual Scale view as appropriate. The taxonomies available are at the top of the main screen. Users can extend the search result based on one of these taxonomies. The system also displays the topics which subsume the user’s query if they exist. In this instance, the topic is “Distributed Artificial Intelligence”.

The user can start navigation from the node by clicking a hyperlink among the sub-concepts or entering a new topic again. Note that the term of the current concept is omitted from each sub-concept to moderate the display space. That is, the sub-concept Agent (4) is the abbreviated form of Artificial intelligence, Agent (4). If we suppose that Data Mining (7) is selected, then the content of the screen will be changed as shown in Figure 6.12. All direct parent and child concepts of the selected concept are displayed.
Figure 6.12 presents the main features of the lattice-browsing interface that shows all direct parent and child nodes of the current concept. To facilitate the user’s understanding of parent and child concepts, we use different colours (red for parents, green for the current concepts and blue for the child concepts).

Users who search for Data Mining under Artificial Intelligence find that there are only 7 researchers in this area. However, this node has 2 parents and so the lattice view makes it obvious that there are in fact 17 researchers in the School who do research in Data Mining. The user can navigate the parent concepts to search for more general documents or navigate the child concepts to get more specific documents. If the user selects a parent concept “Data Mining (17)”, s/he can observe the lattice from the “data mining” point of view.
Figure 6.13. An example of a textword search.

If the entered term does not exist in the concepts of the lattice, a typical textword search is carried out. Documents will be retrieved which contain these textwords in their contents and then a sub-lattice will be constructed with the retrieved documents and their keywords. Figure 6.13 shows an example of this textword search. Navigation is still via the same lattice display, but only accesses the sub-lattice. In this case, the system sends a log file to an engineer so s/he can decide whether more appropriate research topics should be included for the pages. The user can return to the full lattice at any stage via the link Artificial Intelligence in the above screen.

More importantly, a partially ordered hierarchical display is also available. This display is generated using the Conceptual Scale extension to FCA. In essence this gives a view of a lattice formed from objects that have the specified attribute-value pairs. Figure 6.14 shows an example of the nested structure of the concept “Artificial Intelligence”. We build a concept lattice using the result pages with their topics as an outer structure and scale up other attributes into an inner nested structure. The nested structure is constructed dynamically and associated with the current concept of the outer structure. In other words, the nested attribute values are extracted from the result pages.
A nested pop-up menu appears when the user clicks on the “nested” icon in the front of the current node. If the user clicks on one of the menu items, the results will be changed according to the selection. For instance, we suppose that the user selects the menu items “Position” → “Academic Staff” → “Professor”. The result then will be changed as shown in Figure 6.15. Numbers in brackets indicate the number of documents corresponding to the attribute value. For a more detailed discussion of Conceptual scaling for the proposed system refer to Section 5.5 in the previous chapter.

As well, a knowledge engineer can arrange related terms by accessing a tool which allows him or her to set up hierarchical grouping related terms under a common name. Then, when a user’s query is related to the grouping(s), the grouping name is included into the nested structure on the fly. An example can be seen in Figure 6.19 in the following section.
Figure 6.15. The search result with the selection of nested items.

This shows the result of the selection: “Position” → “Academic Staff” → “Professor” in Figure 6.14. The user can read that there are four researchers whose research topic is “artificial intelligence” and whose position is professor.

The user can also extend search results by using one of the imported taxonomies available. In this case the ACM, ASIS&T, Open Directory Project (DMOZ) and the local UNSW taxonomy are available at the top of the search screen. Using one of these will recreate the lattice assuming that any documents annotated with a term from the imported taxonomies also has all the parent terms for that term up the hierarchy. At present the taxonomies are manually imported from the relevant Web pages. As XML representation standards for ontologies become better established, importing a taxonomy and using it to give a different lattice view will only require entering a URL.
Figure 6.16. An example of the search result extended by a taxonomy.

Figure 6.16 shows the result from Figure 6.11 as extended by the ASIS&T taxonomy. Note that the change is the number of AI researchers from 39 (Figure 6.11) to 45 (Figure 6.16). The taxonomy includes the term “artificial intelligence” so that the documents are retrieved annotated with not only the term “artificial intelligence”, but also with child terms of the term “artificial intelligence” in the taxonomy. One can browse this lattice or alternatively one can navigate the lattice without a taxonomy at any stage via the link NONE at the screen.
6.3.2. Domain of Proceedings Papers

Our first implementation of the proposed approach with FCA was a system that gives access to the papers of the on-line Banff Knowledge Acquisition Proceedings. Since 1996, all papers for these proceedings have been published on the Web (KA96, KA98 and KA99; http://ksi.cpsc.ucalgary.ca:80/KA/). The system is accessed at http://pokey.cse.unsw.edu.au/servlets/Search. We have previously described the system implemented on this domain (Kim and Compton 2000; 2001a).

All papers were annotated by a knowledge engineer. The focus of this implementation was to explore some possibilities for a browsing mechanism based on Formal Concept Analysis. However, anyone with access to the WWW can set-up and change annotations to any page on the World Wide Web. That is, there is no security for this domain.

One difference with the system for the domain of research interests is that a hierarchical conceptual clustering of the documents associated with a user’s query is supported as shown on the left side of Figure 6.17. This shows an automatic document-clustering feature similar to general clustering search engines such as Vivisimo and WiseNut. However, clusters here result from relationships between objects (documents) and clusters (organised terms - keywords) based on FCA.

To construct the clustering structure, the system formulates a sub-concept lattice using the retrieved documents and their keywords based on FCA. Then, for each formal concept at the first level of the lattice, a hierarchical clustering is built using their child concepts for three levels of the lattice. This means that the concept lattice is converted into a hierarchical structure. The clusters are dynamically constructed from the search results of the user query. We have tried to provide diverse interaction modes to assist users with different interaction preferences and different needs.

Lattice browsing and other search features such as a textword search, nested hierarchical browsing are also supported as in the system of research interests.
The left-hand side of Figure 6.17 shows a hierarchical clustering structure for the query “ripple down rules”. The search result of the query is displayed on the right-hand of the screen. Sub-clusters of each item will appear in pull-down and pop-up menus when an item is clicked. Users can obtain more specific documents by selecting one of these sub-clusters. Using the link Lattice Browsing at the top of the screen, users can browse the concept lattice to find documents in the same way as the domain of research interests in the previous section. Figure 6.18 shows the browsing scheme based on a concept lattice for this domain.
Figure 6.18. An example of navigating the concept lattice.

This shows the lattice-based browsing interface for the Banff proceeding domain. The features are same as Figure 6.11.

Figure 6.19 shows the nested menu structures for the grouping “ripple down rules”. The nested menu items for “Authors”, “Publication Years” and “Proceeding Titles” were obtained from predefined domain attributes. But the item “RDR” was obtained from the grouping of related values together under a grouping name that a knowledge engineer had previously set up. This is an example of conceptual scaling for a one-valued context presented in Section 5.5.2.
6.4. Chapter Summary

The purpose of the implementations described was to demonstrate and evaluate the proposed approach through a prototype with case studies. The system we have developed was aimed at multiple users being able to add and amend document annotations whenever they chose. The users were also assisted in finding appropriate annotations and the lattice was immediately updated. The end users could find documents both by browsing a lattice-based conceptual structure and by conventional Boolean query. Conceptual scaling for the lattice structure was also supported to allow
users to find more specific results from the interrelationship between specified attribute value pairs and the keywords of documents.

We made the system available on the School Web site and recorded all users’ activities - both for searching and annotating their home pages. We also provided Web evaluation questionnaires on user’s preferences for the lattice-based browsing mechanism and on the efficiency of the annotation mechanisms. The next chapter will present the experimental results on the proposed system based on these activities.

Certainly, it is expected there would be a better response to such a system, where users could change their research topic annotation and immediately see the impact of the change in the clusters of researchers that resulted. However, the objective of the implementation was to observe whether the proposed mechanism could be applied to a document management system for specialised domains.
Chapter 7
Experimental Evaluation

This chapter presents our experiment in using the system that we proposed in Chapter 5. It was used to find staff and student home pages based on their research interests in the School of Computer Science and Engineering, University of New South Wales (UNSW). For the experiment, the system was made available on the School Web site and recorded all users’ activities both for searching, and adding and changing the annotation of their home pages. Evaluation forms were also set up for both lattice-based browsing and annotating which we invited users to fill in.

To date 80 annotated home pages are registered in the system. About 300 search activities were performed both by internal and external users. More use of the system may be required to further evaluate useability. However, we believe that the data we have gathered for this experiment is enough to determine whether the proposed system is a useful alternative for document management and retrieval for a specialised domain. We have previously presented a preliminary evaluation of the system (Kim and Compton 2002a; 2002b).

Section 7.1 gives an overview of the experiment. Section 7.2 presents the experimental results. Firstly, we present the results on whether the annotation mechanisms gave users useful assistance in annotating their home pages so that the search performance of the system was improved. These come from an analysis of the users’ annotation activities that we logged, as well as an analysis of the questionnaire data. Secondly, the necessity of document management systems that evolve, rather than systems that only use a priori ontologies (or taxonomies), is discussed in relation to the experiments. Thirdly, we present the results on whether the browsing structure evolved into a reasonable consensus when users freely annotated their documents.
7.1. Experimental Design

Our first implementation of the proposed approach with FCA was a system for the papers of the on-line Banff Knowledge Acquisition Proceedings as presented in Section 6.3.2. With the domains of papers, gathering users’ statistics would have been extremely difficult because we do not have the control over the Banff server. We chose the annotation of researchers’ home page as an evaluation domain because it was anticipated that researchers would be motivated to assist prospective research students and other collaborators to find them. Home pages generally describe research interests and the system would help students in finding interesting research areas. In addition, as there are always students looking for supervisors it is anticipated that in time there would also be sufficient browsing, and sufficient prospective students would be willing to fill out the Web evaluation questionnaire.

It was also felt that the researchers might be more interested in using the system if they could immediately see where they fitted in the existing lattice, and so they may be more motivated to make changes. To this end, the starting lattice was populated by automatically annotating researchers’ home pages (37 academics) with terms specified as their research areas in the School’s research topic index. The problem with this approach of course is that we cannot then see how a lattice would evolve starting from scratch. Once the system was set up, it was opened up for use by staff, research fellows and Ph.D. students. In this case their home pages were not initially annotated.

Previously the School had used simple research topic indices and permuted indices\textsuperscript{73}. These require the School office to make periodic requests for lists of topics and individuals respond independently. The School now has links through to the lattice-based browser from a number of different pages\textsuperscript{74}. However, as this is an experimental project the various School research indices have been continued and the links to the lattice search have not been particularly highlighted.

\textsuperscript{73} http://www.cse.unsw.edu.au/school/research/curresearch.html and http://www.cse.unsw.edu.au/school/research/research2.html.

7.2. Experimental Results

Presently 80 annotated home pages are registered in the system with an average of 8 research topics (ranging from 2 to 27). The lattice contains 471 nodes with an average of 2 parents per node (ranging from 1 to 10) and path lengths from 2 to 7 edges.

Table 7.1 shows the number of home pages annotated. Of the 37 academics who had their home pages automatically annotated, 16 refined their research topics after deployment. This means that 21 researchers among the 37 might be either happy with their automatic annotations or ignored the experiment. After the system was made available on the Web, another 43 research staff and students carried out annotation of their pages. Consequently, 59 staff and students have actively participated in the annotation of home pages. Of interest, almost half of the staff and students who started out with the system later changed their annotations. This result alone suggests the need for evolutionary user based annotation.

<table>
<thead>
<tr>
<th></th>
<th>Pages automatically annotated before deployment</th>
<th>Pages annotated by research staff or students</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of annotated pages</td>
<td>37</td>
<td>43</td>
<td>80</td>
</tr>
<tr>
<td>Number of pages for which the initial annotations were later changed</td>
<td>16</td>
<td>19</td>
<td>35</td>
</tr>
</tbody>
</table>

7.2.1. Annotation Mechanisms

This section presents the results on the annotation mechanisms for incremental development. The main difference between the system we have implemented and previous work in this area is an emphasis on incremental development and evolution for specialised domains. These come from an analysis of the users’ annotation activities that we logged and an analysis of the questionnaire data.

7.2.1.1. Users’ Annotation Activities

The results of the users’ annotation activities will be presented based on each phase of the annotation process described in Chapter 5. Table 7.2 summarises those phases.
Table 7.2. Task for each phase of the annotation process.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>Select topics from list or add new topics. The list includes topics used by other researchers (and can be viewed by researchers) and topics from external taxonomies.75</td>
</tr>
<tr>
<td>Phase 2</td>
<td>Select topics from terms suggested from taxonomies which are relevant to the assigned topics in Phase 1</td>
</tr>
<tr>
<td>Phase 3</td>
<td>Select topics from the terms that co-occur in the lattice with the assigned topics in Phase 1</td>
</tr>
<tr>
<td>Phase 4</td>
<td>Adding topics to differentiate home pages</td>
</tr>
<tr>
<td>Phase 5</td>
<td>Adding new terms</td>
</tr>
<tr>
<td>Phase 6</td>
<td>Logging users’ queries</td>
</tr>
</tbody>
</table>

Table 7.3. Number of terms added at each phase for 59 home pages.

<table>
<thead>
<tr>
<th>Phase</th>
<th>A number of assigned terms</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1: Reused terms in the system or new added terms</td>
<td>468</td>
<td>79 %</td>
</tr>
<tr>
<td>Phase 2: Imported terms from taxonomies</td>
<td>19</td>
<td>3.2 %</td>
</tr>
<tr>
<td>Phase 3: Co-occurred terms in the lattice</td>
<td>99</td>
<td>16.8 %</td>
</tr>
<tr>
<td>Phase 4: Added terms to distinguish pages</td>
<td>2</td>
<td>0.4 %</td>
</tr>
<tr>
<td>Phase 5: Added terms from new terms</td>
<td>2</td>
<td>0.4 %</td>
</tr>
<tr>
<td>Phase 6: Added terms from users’ queries</td>
<td>1</td>
<td>0.2 %</td>
</tr>
<tr>
<td>Total</td>
<td>591</td>
<td>100 %</td>
</tr>
</tbody>
</table>

Table 7.3 shows the number of terms added at each phase for the 59 pages which were annotated by the active participation of researchers. There were 99 annotation activities performed by researchers: 43 for new annotation cases and 56 changes to existing annotations (including 16 cases where the pages had been populated prior to use). Note that there were 56 changes to 35 home pages. Therefore, some of these pages were changed more than once (< 56 refinement activities). The number of assigned terms for each phase was recalculated for the home pages that had been changed by referring to the annotation histories. Then, the total number of assigned terms for the 59 pages at Phase 1, 2 and 3 were computed.

75 Note that two taxonomies have been imported from commonly available Web sites (i.e., ACM and ASIS&T) and adjusted for our purpose by pruning. We have also developed a hybrid taxonomy called UNSW by combining a number of taxonomies which are considered relevant to “research topics” areas for the School of Computer Science and Engineering (CSE).
The results indicate that 19 terms (3.2%) were discovered from the imported taxonomies and that 99 terms (16.8%) were co-occurring terms that were suggested. Two terms were discovered for distinguishing related documents, and another two terms from adding new terms and one term from the data that we logged from users’ query activities. This means that 123 terms (21%) of the total 591 were provided by the supplementary supported mechanisms.

Phase 1, 2 and 3
As presented in Chapter 5, when a user starts to annotate their home page, the system displays all the topics used by others as well as all those contained in the taxonomies (Phase 1). After the initial assignment, the user can view other terms that are imported from the taxonomies (Phase 2). These terms are all parents of the terms the user entered in Phase 1 from the taxonomies. In Phase 3, the user can also view terms that co-occur in the lattice with the terms s/he has provided at Phase 1. The user can then annotate their home page with some of these terms if desired.

Phase 4: Adding Topics to differentiate Home Pages
This annotation method is initiated when one user annotates his/her home page with the same topics as another user. Note that this feature is imported from the RDR technique which differentiates between a new case being added and the other stored cases. The system retrieves a document that is annotated with the same terms and suggests that the user may wish to differentiate their home page from the document. Only two such cases occurred. This mechanism may have been more important if there had been no start up annotations. Note that 37 academics’ home pages were annotated automatically when the system was made available to users. This method might be useful for two researchers who immediately decided they wanted to distinguish themselves.

Furthermore, this will be more significant in a domain which has a densely similar context (i.e., a context which has many similar documents). We have observed that in document management for an individual using the proposed approach\textsuperscript{76}, this situation for differentiating related documents occurred quite often.

\textsuperscript{76} http://pokey.cse.unsw.edu.au/servlets/DMR?kb=mihyek/bookmark&userId=mihyek.
Phase 5: Adding New Terms

The additional annotation method provided is relevant when a new term is entered for a new document and the term may also appropriately apply to other documents already in the system. This situation has occurred in 240 cases, but only 25 cases among those were considered. A knowledge engineer examined the cases suggested by the system (240 cases) and interviewed the annotators of 25 of the pages to decide whether the new term is relevant to these home pages. As a result, two researchers added the suggested term to their research topics. We have not paid close attention to this mechanism. We realised that it was too costly to go through every case and it was also difficult for the knowledge engineer to determine whether the new term is relevant to the suggested pages. However, our aim was to consider as many factors as possible that may be useful for discovery of terms relevant to the annotated documents.

Phase 6: Logging Users’ Queries

Further mechanisms involved referring cases to a knowledge engineer when a query did not include the topics used by annotators and a textword search was invoked. A log of such transactions was sent to the knowledge engineer. This phase is caused by the users’ search activities, rather than the users’ annotation activities. However, this is also one of the mechanisms for the incremental development of the system.

There were 161 such cases. For five cases (“agent oriented systems”, “compiler design”, “mobile commerce”, “logistics” and “compilers”) the knowledge engineer interviewed researchers whose home page contained the terms. As a consequence, the term “compiler” was added as a research topic to a home page. Other terms (156 terms) were regarded as not relevant. Among those, 59 terms were related to the researchers’ name (first name or last name and both). This suggests that it may be useful to index documents with the names of researchers. However, in this application there were adequate mechanisms to find researchers by name. There were also five abbreviation terms: “ai”, “db”, “b2b”, “ICMS” and “cs”. It was clear that abbreviations would enhance the retrieval processes. As a consequence, the (rather obvious) abbreviation synonyms shown in Table 7.4 were added to the system using the ontology editor presented in Chapter 6.
Table 7.4. Examples of abbreviation classes registered to the system.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full word</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>B2B</td>
<td>Business to Business</td>
</tr>
<tr>
<td>DB</td>
<td>Databases</td>
</tr>
<tr>
<td>DBMS</td>
<td>Databases Management Systems</td>
</tr>
<tr>
<td>E Business, E-Business</td>
<td>Electronic Business</td>
</tr>
<tr>
<td>E Commerce, E-Commerce</td>
<td>Electronic Commerce</td>
</tr>
<tr>
<td>FCA</td>
<td>Formal Concept Analysis</td>
</tr>
<tr>
<td>HCI</td>
<td>Human Computer Interaction</td>
</tr>
<tr>
<td>KA</td>
<td>Knowledge Acquisition</td>
</tr>
<tr>
<td>KBS</td>
<td>Knowledge Based Systems</td>
</tr>
<tr>
<td>MCRDR</td>
<td>Multiple Classification Ripple-Down Rules</td>
</tr>
<tr>
<td>NRDR</td>
<td>Nested Ripple Down Rules</td>
</tr>
<tr>
<td>OS</td>
<td>Operating Systems</td>
</tr>
<tr>
<td>RDR</td>
<td>Ripple-Down Rules</td>
</tr>
<tr>
<td>SE</td>
<td>Software Engineering</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>

The remainder (92 terms) appeared to be tests of the system’s retrieval mechanisms rather than identifying research areas. They included terms such as “delphi”, “singapore”, “help”, “topic” and others. Some users seemed to regard the system as a general search engine developed to support finding information within the School Web sites. This seemed to be confirmed by terms such as “summer project”, “java”, “unix primer”, “jdk”, “telnet”, “linux”, “scholarship” and others.

**Other observations from the users’ annotation activities**

Another observation from the annotation activities was that 88.1% of the annotators (52 of 59 those who carried out annotation) promptly confirmed the effect of their annotations by browsing the lattice. Twenty-one annotators (40.4%) changed the assigned terms after viewing the browsing structure. Overall, 35 home pages underwent further changes. It is worth noting that we did not give any detailed explanation of the
annotation procedures in advance. It was simply advertised that the system was available, and told users the system would enable them to annotate their home pages with a set of topics. An interesting observed behaviour was that research students examined the annotated research topics of their supervisors or collaborators who belonged to the same research group and then selected some of these topics. The selected topics were usually more general and described the research group overall.

The reusability of terms should be noted. The 59 researchers who actively annotated their home pages used 591 terms. 550 (93%) of these were terms that were supported by the system, while 41 (7%) of the terms were newly entered into the system. This is a very high level of reuse and suggests that the support available to assist annotation must have been useful.

In summary, the tools available to the annotators provided a good level of useful assistance. Considering that the users received no training or even no advice on how the system worked and what support was available, we feel this evaluation provides strong support for the value of the approach. The tools available to the supervising knowledge engineer were less useful, but did provide some help.

7.2.1.2. Survey: Questionnaire on the Annotation Mechanisms

This section presents the evaluation data of the annotation mechanisms from an on-line Web-based survey. Slaughter et al. (1994) examined the effectiveness of on-line questionnaires and reported that on-line surveys were as good as paper and pencil surveys. Harper et al. (1997), and Kuter and Yilmaz (2001) also addressed the characteristics of Web-based questionnaires. Perlman (1997) and Rho (2001) developed some examples of Web-based questionnaires. Following these studies, the questionnaire for the annotation mechanisms was designed (Figure 7.1). The questionnaire was implemented using standard HTML forms that let users click on radio buttons and enter comments into text areas. The questionnaire was linked to the search pages of the system. Thirty-seven questionnaires were completed (63% of the 59 participants).
Questionnaire on the Annotation Mechanisms on Your Research Topics

Initial idea on the questionnaire format from: Perlman G., 'Web-based User Interface Evaluation with Questionnaires.'

This tool has been developed as part of Miluye Kim's PhD project, so we would be very grateful if you would take a few minutes filling in the following evaluation.

Many thanks,

Miluye Kim and Paul Compton

Q1 How was the annotation mechanism to use? (1) Overall, it was easy to annotate my research topics difficult easy (2) Overall, it was helpful in defining my research topics unhelpful helpful

Q2 Listed research topics (1) Are there too many topics on the list? too few too many (2) Are they at the right level of generality? too specialized too general (3) Are they appropriate? inappropriate appropriate (4) Were they helpful for annotating your research topics? unhelpful helpful

Your comment - any other comments you can make would be greatly appreciated

Figure 7.1. Questionnaire used for the annotation mechanisms.
Table 7.5 shows the questionnaire results on Q1 “How was the annotation mechanism to use?”. Users were able to express their opinions on a 5-point Likert scale for each question. The majority (95%) of the respondents characterised the annotation mechanisms as easy to use, selecting 4 or 5 on the 5-point scale. No one rated 1 or 2 for this question. The mean score for this question was 4.43.

Regarding the helpfulness of the annotation mechanisms, 75.5% of the respondents considered that the mechanisms were helpful. Only 3% indicated the mechanisms as being not really helpful. The average score for this question was 4.03.

These results are consistent with the results on the logs of the users’ annotation activities presented in the previous section. Conclusions therefore can be made that the annotation mechanisms were easy to use and were helpful in defining annotators’ research topics.

One respondent, who rated Question1, part 2 as 2 (i.e., unhelpful), commented that a huge list of supported topics was daunting. The respondent suggested that a restricted list of keywords - automatically extracted from the home page - should be presented to the users. This issue will be discussed again with the results of Question2.

<table>
<thead>
<tr>
<th>Q1. How was the annotation mechanism to use?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Overall, it was easy to annotate my research topics</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>(2) Overall, it was helpful in defining my research topics</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Note: (1) $\bar{x} = 4.43$  (2) $\bar{x} = 4.03$. 

145
Table 7.6 presents the questionnaire results on the supported topics. The aim of these questions was to observe whether the topics provided were appropriate and helpful for users’ annotating, and whether they were at the right level of generality. In addition, it is examined whether the mechanism which simply listed topics used by other researchers in a long list was adequate or whether a more efficient method is needed.

**Table 7.6.** The questionnaire results on the research topics supported.

<table>
<thead>
<tr>
<th>Question</th>
<th>Categories</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2. Listed research topics</td>
<td>too few</td>
<td>0</td>
<td>1</td>
<td>15</td>
<td>18</td>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>too many</td>
<td>0%</td>
<td>2.5%</td>
<td>40.5%</td>
<td>49%</td>
<td>8%</td>
<td>100%</td>
</tr>
<tr>
<td>(1) Are there too many topics on the list?</td>
<td>too specialised</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>too general</td>
<td>0</td>
<td>2</td>
<td>27</td>
<td>7</td>
<td>1</td>
<td>36</td>
</tr>
<tr>
<td>(2) Are they at the right level of generality?</td>
<td>inappropriate</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>25</td>
<td>5</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>appropriate</td>
<td>0%</td>
<td>0%</td>
<td>19%</td>
<td>67.5%</td>
<td>13.5%</td>
<td>100%</td>
</tr>
<tr>
<td>(3) Are they appropriate?</td>
<td>unhelpful</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>helpful</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>22</td>
<td>13</td>
<td>35</td>
</tr>
<tr>
<td>(4) Were they helpful for annotating your research topics?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: (1) $\bar{x} = 3.62$ (2) $\bar{x} = 3.19$ (3) $\bar{x} = 3.95$ (4) $\bar{x} = 4.3$.

Twenty-one participants (57%) indicated that there were too many topics on the list with another 40.5% giving a neutral response. One participant rated 2 (i.e., few) for the question. This respondent might think that there were not many topics which covered his/her particular research areas. The mean score for this question was 3.62, which is to the “many” side of neutral. As for the generality of the research topics, 73% of the respondents gave a neutral response. Relating to the appropriateness of the terms, 81% indicated that the topics were appropriate. Similarly, 94.5% of the respondents characterised the listed topics as helpful for annotating their research topics.

Table 7.7 shows the cross distribution of the responses about the number of topics on the list and their generality. Of those who regarded the number of listed topics as many (i.e., rated as 4 or 5), 66.5% gave a neutral response to the level of generality and 24%
thought the topics are too general. Only 9.5% indicated the topics as too specialised. This seems to show that the “long length” of the listed topics is due to the wide range of research areas in the School, rather than a high level of specialisation.

The natural tendency for each researcher is to have a slightly different research profile. More general topics can be shared, but more specific terms can be brought in by each researcher to distinguish themselves. An observation from the users’ annotation activities that we logged, indicated that all users examined the list of topics first before adding new topics which did not exist in the list. The newly entered terms were usually more specific and oriented to the annotator. A possible factor affecting this question is that annotators can have different opinions depending on previous annotations. When one annotates his/her home page after a number of collaborators have already annotated their pages, the annotator can think that the supported topics are quite specialised or have an appropriate level of generality. Earlier, the annotator might feel the topics provided are too general, as the only relevant topics available are general.

**Table 7.7.** Cross-distribution between the number of topics on the list and their generality.

<table>
<thead>
<tr>
<th>Level of generality</th>
<th>(1) Too few</th>
<th>(2) Level of generality</th>
<th>(1) Too many</th>
<th>N (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>too specialised</td>
<td>too many</td>
<td>too many</td>
<td>too many</td>
<td>N (2)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>N (1)</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 7.8 shows the cross distribution of the responses about the number of topics on the list and their appropriateness. Of those who indicated the number of listed topics as many (i.e., rated as 4 or 5), 81% thought the topics are appropriate and 19% gave a neutral response. Of those who gave a neutral response for the listed topics, 87% indicated the topics are appropriate. This seems to show that the terms are not inappropriate even when the number of listed topics was seen as large.
Table 7.8. Cross-distribution between the number of topics on the list and their appropriateness.

<table>
<thead>
<tr>
<th>(3) Listed topics appropriate?</th>
<th>(1) Topic on the list</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>N (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>inappropriate</td>
<td>too few</td>
<td>1</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>too many</td>
<td>2</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0</td>
<td>0%</td>
<td>1</td>
<td>100%</td>
<td>2</td>
</tr>
<tr>
<td>appropriate</td>
<td></td>
<td>4</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 7.9 shows the cross distribution of the responses for appropriateness and helpfulness of the listed topics. Those who regarded the listed topics as appropriate also indicated the topics were helpful. Of those who gave a neutral response for appropriateness, 71% thought the listed topics were helpful (i.e., rated as 4 or 5). The respondents generally gave a higher rank to helpfulness than appropriateness.

Table 7.9. Cross-distribution between appropriateness and helpfulness of the listed topics.

<table>
<thead>
<tr>
<th>(4) Were they helpful</th>
<th>(3) Are the listed topics appropriate?</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>N (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>helpful</td>
<td>inappropriate</td>
<td>1</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>appropriate</td>
<td>2</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N(3)</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0%</td>
<td>7</td>
</tr>
</tbody>
</table>

Therefore, it can be said that the listed terms were appropriate with a reasonable level of generality and helpful for annotating users’ research topics, even though there were too many topics on the list to go through.

The list of topics had around 225 terms. Going through every topic in this list to chose one’s research topics can definitely be time-consuming. To overcome this weak point, the system supported a function to select topics from other researcher’s topics. In other
words, an annotator can choose other annotated researchers. Then, the annotator can select topics from lists of topics based on these selected researchers. The researchers may be the annotator’s collaborators or supervisors with whom the annotator may share interests. With this function, to some extent the annotator can moderate the number of topics to be considered. This may account for the 40.5% of respondents who rated the topic list as 3 (i.e., reasonable). It can be noted that although selecting from other researchers may limit the choices available, further phases of the annotation process show other related topics.

However, as one respondent noted in the comment for Question1, a more efficient mechanism for considering topics needs to be explored. It would be better to support a mechanism that displays terms extracted from the annotator’s home page first. In other words, when a researcher annotates his/her home page, terms could be extracted from their home page and suggested first. The extraction of relevant terms to a page could be done using some machine learning techniques.

In summary, not only were the annotation mechanisms easy to use and helpful, but also the terms available in the system were helpful in defining annotators’ research topics. However, a more efficient tool is required to shorten the list of available topics, rather than having a long list showing all topics available.

### 7.2.2. Ontology Evolution

One goal of this thesis is to explore the possibilities of document annotation systems that do not commit to *a priori* ontologies. The aim is to develop techniques for assisting users in annotating a document as an ontology evolves. Instead of defining the ontology from the outset, we would like the system to assist the users to make extensions to the developing ontology so that the ontology is improved.

Recall that we have imported two taxonomies ACM and ASIS&T, and developed a taxonomy called UNSW by combining the research areas at the School Web sites and a number of taxonomies which are considered relevant to the School research areas.
One of the key components of the proposed approach is to show annotators all the parents of any terms that occur in the imported taxonomies. Then, users determine how relevant the proposed terms are to their documents and select any combination of superclass terms to be added to their documents. A key component of the annotation was in suggesting key terms and users freely selecting them without considering any hierarchies of the terms.

The critical advantage of this is that terms that seem too general, even if only part of the way up the hierarchy, can be omitted. The user does not have to consider if the terms are too general, in fact the parent-child relations are not indicated - a simple list of terms is shown. The result of this is a new taxonomy that is made up of parts of other taxonomies that users perceive as most useful along with other terms they add. We believe that this may provide a very simple but powerful way of validating and improving on the ontological standards that are being established.

Table 7.10 shows the use of the imported terms. Of the 207 terms suggested for 59 cases (researchers), 19 were used for annotation. The annotators were interviewed to investigate the reasons why the other suggested terms were not selected. The most common response was that the proposed topics were too general and irrelevant in specifying their research area, even though they were applicable.

<table>
<thead>
<tr>
<th></th>
<th>A number of terms</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total suggested taxonomy terms</td>
<td>207</td>
<td>-</td>
</tr>
<tr>
<td>Selected terms</td>
<td>19</td>
<td>9.2 %</td>
</tr>
<tr>
<td>Non-selected terms</td>
<td>188</td>
<td>90.8 %</td>
</tr>
</tbody>
</table>

It can be assumed that the imported terms would be appropriate terms to use, particularly since the taxonomies imported (i.e., ACM and ASIS&T) would seem to be appropriate for a school of Computer Science and Engineering (CSE). Therefore, if it is supposed that the taxonomies represent an a priori taxonomy of the research areas of the School, then the percentage of the selected terms should be high.
However, only 19 of the 207 terms suggested were used for annotation. Recall that the 207 terms suggested (for 59 cases) are all parent terms that occur in the taxonomies for the terms assigned by the user at the first stage. Note that these 59 cases actively annotated their home pages. Some terms that occur in the taxonomies can be selected when users annotate their topics at the first stage as terms in the taxonomies are also supported at the stage. Thus, if we calculate the ratio of the terms of the taxonomies used for annotation, the percentage of the selected terms from the taxonomies will definitely increase. However, an important fact is that most of the general terms suggested were not selected.

The details of the relevance of terms suggested from taxonomies and the consequent retrieval of documents (researchers) are shown in Table 7.11. Here the Open Directory Project, which is regarded as one of the world’s biggest human-edited taxonomies, has also been included. But this was not available during annotation.

Recall that the lattice shows all the researchers who use a particular term, and that this number is increased by importing terms from taxonomies and considering that pages are implicitly annotated by any terms in the taxonomy that are parents of terms selected by the researcher. These would be the results if the researchers were obliged to conform to that ontology (policy).

Table 7.11 shows the number of researcher home pages retrieved using the various terms in the left column with and without imported taxonomies. The first retrieval column shows the number of pages retrieved using only the researchers own annotation of their home pages as shown in the lattice. The remaining columns show the retrieval when it is assumed that any parent terms in the various taxonomies should also apply to pages that are annotated with their child terms. A hyphen “-” indicates that the term on the left-hand side is not contained in the relevant taxonomy. Some terms included in Table 7.11 have no children in the respective taxonomies so that the numbers of pages retrieved are the same. These terms are marked with an asterisk “*”. This is to show the existence of the terms in the appropriate taxonomies. Figure 7.2 shows some partial hierarchies of the taxonomies for the terms used in Table 7.11.
Table 7.11. Document retrieval using various taxonomies.

<table>
<thead>
<tr>
<th>Terms</th>
<th>Number of researcher home pages retrieved</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lattice Only</td>
</tr>
<tr>
<td>Artificial Intelligence</td>
<td>39</td>
</tr>
<tr>
<td>Knowledge Engineering</td>
<td>3</td>
</tr>
<tr>
<td>Knowledge Representation</td>
<td>18</td>
</tr>
<tr>
<td>Knowledge Management</td>
<td>4</td>
</tr>
<tr>
<td>Knowledge Discovery</td>
<td>7</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>24</td>
</tr>
<tr>
<td>Learning</td>
<td>5</td>
</tr>
<tr>
<td>Information Processing</td>
<td>1</td>
</tr>
<tr>
<td>Information Retrieval</td>
<td>10</td>
</tr>
<tr>
<td>Internet</td>
<td>4</td>
</tr>
<tr>
<td>Databases</td>
<td>11</td>
</tr>
<tr>
<td>Computer Programming</td>
<td>1</td>
</tr>
<tr>
<td>Programming Languages</td>
<td>4</td>
</tr>
<tr>
<td>Knowledge Acquisition*</td>
<td>19</td>
</tr>
<tr>
<td>Spatial Representation*</td>
<td>3</td>
</tr>
<tr>
<td>Data Mining*</td>
<td>17</td>
</tr>
<tr>
<td>World Wide Web*</td>
<td>5</td>
</tr>
</tbody>
</table>

In Table 7.11, we can observe that not only do the ACM and ASIS&T taxonomies and the Open Directory have very different ideas of what constitutes “Knowledge Engineering”, but also that the ACM and ASIS&T taxonomies have a different scheme on “Machine Learning” and “Learning”. The ACM taxonomy and the Open Directory do not use the term “Knowledge Engineering”. Furthermore, the terms “Knowledge Management” and “Knowledge Discovery” are not used in the ACM and ASIS&T taxonomies. However, there is obviously a high degree of consistency with terms such as “Information Retrieval” and “Programming Languages” and “Knowledge Representation”. The clustering of the term “Databases” is highly consistent between the ACM and ASIS&T taxonomies. However, the Open Directory has more terms classified under the term “Databases” that caused a big difference in the number of retrieved pages compared with other taxonomies. “Data Mining” is a representative example of these sub-terms.
These phenomena not only suggest random variations, but specific and relatively consensual decisions about the value of the various terms available. There can be a commitment to use some particular taxonomy, but there is no best structure. It therefore seems highly advantageous to allow users in various communities very flexible access to such ontological resources, so the most appropriate use for the community can emerge.

Next, we would like to examine whether retrieval performance can be enhanced when a user’s query uses the taxonomies since this is closely interrelated with the issue of how knowledge structures should evolve over time. Traditional information retrieval has often incorporated the use of pre-defined classification systems, thesauri or taxonomies. A lattice-based model for information retrieval has also been associated with the use of domain thesauri presenting an improvement in retrieval efficiency (Carpineto and Romano 1996a; Cole and Eklund 1996a).
However, the problem is that inheritance in taxonomy hierarchies for instances (objects) is often not transitive. When a taxonomy or thesaurus is constructed, the most appropriate thesaural entries for representing documents are selected and the entry terms are organised in hierarchies. Hence, the inheritances in the hierarchies are not always transmittable when the taxonomy is instantiated with objects.

In other words, the problem with a fixed hierarchical structure is that there may be no “right place” for a document. Let us look at the term “Data Mining” which is clustered both under the terms “Artificial Intelligence” and “Databases” in the Open Directory Project in Figure 7.2. Sometimes it is not clear where to place a document about “Data Mining”, but is also about both “Artificial Intelligence” and “Databases”. Alternatively, there may be a document about “Data Mining”; which does not belong with either “Databases” or “Artificial Intelligence”, but belongs with graph theory with a fixed taxonomy or thesaurus, which will be stored inappropriately.

We examined retrieval performance on queries that are relevant to the taxonomical terms in Table 7.11. McGuinness (2000) noted that authors often became highly literate in the domains they are involved in. Therefore, believing that authors (annotators) are the most appropriate agents to assign concepts for their documents, we assumed that the performance of a search result with a topic adapted by researchers in the lattice is complete in precision77. Based on this assumption, we computed retrieval performance on the queries in Table 7.11. The results indicated that average retrieval performance with the taxonomies was low in precision (an average decrease in precision of 0.35 - see Appendix 1) than the lattice retrieval performance. Of course, this is only for search terms that exist in the taxonomies so that the average decrease in precision can be less.

However, this result can suggest that it should not be assumed that a reasoning mechanism along with the hierarchies of a taxonomy can always enhance retrieval performance. It requires a more precise consideration of the involved domain.

77 Precision is the ratio of relevant documents retrieved for a given query over the total number documents retrieved.
In summary, users’ selection of the terms suggested from the imported taxonomies has been examined. Additionally, retrieval performance for the terms in the taxonomies was examined. We believe that the experimental results confirm our view of how a knowledge structure of concepts (or ontology) for a domain should be evolved with emphasis on the significance of context. However, as standards for representing ontologies take hold, these small community systems will be able to very flexibly import ontologies and make selective use of their resources.

7.2.3. Lattice-based Browsing

A key difference between the proposed approach and the general information retrieval approach is in the method of clustering documents for browsing. This is based on a lattice model. As indicated earlier, users themselves annotate their home pages in whichever way they like, assisted by the supporting annotation mechanisms. Formal Concept Analysis then generates a conceptual hierarchy for browsing by finding all possible formal concepts which reflect a certain relationship between the annotated terms and home pages. The structure is based on a lattice scheme which forms a multi-parent relationship. The system then updates the concept lattice whenever a new home page is added with a set of topics, or the topics of existing pages are refined.

A key question is whether the browsing structure can evolve into a reasonable consensus when multiple users freely annotate documents. The contrast with other ontology work is that the consensus will emerge rather than being imposed by some groups who have decided what it should be.

However, an issue here is how the emerged structure (or ontology) can be validated. We felt this is an extremely difficult task because there is no best structure for a particular domain and no guidelines are apparent in the literature for evaluating the efficiency of an ontology or taxonomy. This would also be an important issue in ontology communities. As a consequence, we have designed questionnaires and conducted a survey to evaluate the efficiency of lattice-based browsing from the users’ point of view. If the search performance of the system results in efficiency, it may be concluded
that the evolved structure is well organised as a consensus or vice versa. The survey results will be presented in Section 7.2.3.2.

Before the survey results, we would like to present what the browsing structure constructed by multiple users looks like, and what kinds of advantages a lattice-based structure has over a hierarchical structure.

7.2.3.1. Browsing Structure

Figure 7.3(a), (b), and (c) show examples of the browsing structure presented in a flat form. Recall that the 80 home pages of academic and research students have been registered with an average of 8 research topics. The concept lattice contains 471 nodes with an average of 2 parents per node and path lengths ranging from 2 to 7 edges. The lattice is continuously evolving as incremental changes are made. The positive survey results on the annotation mechanisms suggest that the browsing structure is organised into a reasonable consensus.

=================================================================

Example 1: Root (80)

<table>
<thead>
<tr>
<th>Category</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>6</td>
</tr>
<tr>
<td>Algorithms</td>
<td>6</td>
</tr>
<tr>
<td>Algorithm Design</td>
<td>3</td>
</tr>
<tr>
<td>Artificial Intelligence</td>
<td>39</td>
</tr>
<tr>
<td>Belief</td>
<td>4</td>
</tr>
<tr>
<td>Clustering</td>
<td>3</td>
</tr>
<tr>
<td>Compilation</td>
<td>2</td>
</tr>
<tr>
<td>Compiler Construction</td>
<td>2</td>
</tr>
<tr>
<td>Compiler Technology</td>
<td>2</td>
</tr>
<tr>
<td>Computational Algebra</td>
<td>2</td>
</tr>
<tr>
<td>Computational Geometry</td>
<td>4</td>
</tr>
<tr>
<td>Computer Architecture</td>
<td>2</td>
</tr>
<tr>
<td>Computer Graphics</td>
<td>3</td>
</tr>
<tr>
<td>Data Mining</td>
<td>17</td>
</tr>
<tr>
<td>Data Structures</td>
<td>3</td>
</tr>
<tr>
<td>Databases</td>
<td>11</td>
</tr>
<tr>
<td>Database Applications</td>
<td>8</td>
</tr>
<tr>
<td>Distributed Computing</td>
<td>5</td>
</tr>
<tr>
<td>Distributed Systems</td>
<td>5</td>
</tr>
<tr>
<td>Electronic Commerce</td>
<td>12</td>
</tr>
<tr>
<td>Formal methods</td>
<td>4</td>
</tr>
<tr>
<td>Functional Programming</td>
<td>4</td>
</tr>
<tr>
<td>Human Computer Interaction</td>
<td>5</td>
</tr>
<tr>
<td>Image Processing</td>
<td>5</td>
</tr>
<tr>
<td>Information Extraction</td>
<td>2</td>
</tr>
<tr>
<td>Information Retrieval</td>
<td>10</td>
</tr>
<tr>
<td>Internet</td>
<td>4</td>
</tr>
<tr>
<td>Knowledge Acquisition</td>
<td>19</td>
</tr>
<tr>
<td>Knowledge-Based Systems</td>
<td>9</td>
</tr>
<tr>
<td>Knowledge Discovery</td>
<td>7</td>
</tr>
<tr>
<td>Knowledge Management</td>
<td>4</td>
</tr>
<tr>
<td>Knowledge Representation</td>
<td>18</td>
</tr>
<tr>
<td>Logics</td>
<td>9</td>
</tr>
<tr>
<td>Logics Programming</td>
<td>8</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>24</td>
</tr>
<tr>
<td>Natural Language Processing</td>
<td>6</td>
</tr>
<tr>
<td>Network Management</td>
<td>2</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>7</td>
</tr>
<tr>
<td>Object oriented Design</td>
<td>2</td>
</tr>
<tr>
<td>Ontologies</td>
<td>5</td>
</tr>
<tr>
<td>Parallel computing</td>
<td>5</td>
</tr>
<tr>
<td>Pattern Recognition</td>
<td>4</td>
</tr>
<tr>
<td>Personalisation</td>
<td>3</td>
</tr>
<tr>
<td>Program Analysis</td>
<td>2</td>
</tr>
<tr>
<td>Programming Languages</td>
<td>4</td>
</tr>
<tr>
<td>Robotics</td>
<td>12</td>
</tr>
<tr>
<td>Semantic Web</td>
<td>6</td>
</tr>
<tr>
<td>Software Engineering</td>
<td>7</td>
</tr>
<tr>
<td>Spatial Reasoning</td>
<td>3</td>
</tr>
<tr>
<td>Text Mining</td>
<td>6</td>
</tr>
<tr>
<td>Web Services</td>
<td>6</td>
</tr>
<tr>
<td>Workflows</td>
<td>6</td>
</tr>
<tr>
<td>World Wide Web</td>
<td>4</td>
</tr>
<tr>
<td>XML</td>
<td>6</td>
</tr>
</tbody>
</table>

=================================================================

**Figure 7.3(a):** Examples of the browsing structure that evolved.

This shows the top-level concepts of the lattice constructed by FCA. Numbers in parenthesis indicate a number of objects which satisfy the term.
**Example 2**: Root (80) => Artificial Intelligence (39)  
Agent Theory (4)  
Agent (4)  
Cognitive Modelling (5)  
Cognitive Robotics (5)  
Combinatorial Algorithms (2)  
Data Mining (7)  
Electronic Commerce (3)  
Image Processing (3)  
Information Retrieval (5)  
Knowledge Acquisition (16)  
Knowledge-Based Systems (8)  
Knowledge Discovery (6)  
Knowledge Representation (14)  
Learning (5)  
Machine Learning (20)  
Mobile agent (3)  
Natural Language Processing (5)  
Neural Networks (5)  
Ontologies (4)  
Pattern Recognition (3)  
Philosophy (5)  
Planning (2)  
Quantum computing (1)  
Robotics (10)  
Spatial Reasoning (2)  
Spatial Representation (3)  
Text Mining (4)  

**Example 3**: Root (80) => Artificial Intelligence (39) => Knowledge Representation (14)  
**Parent Topics**:  
Artificial Intelligence (39)  
Knowledge Representation (18)  
**Sub Topics**:  
Agent Theory (3)  
Agent (3)  
Belief Revision (7)  
Causal Reasoning (4)  
Knowledge Acquisition (6)  
Knowledge Discovery (3)  
Logics (7)  
Machine Learning (5)  
Multi-agent systems (3)  
Nonmonotonic reasoning (7)  
Ontologies (2)  
Robotics (2)  
Theory Revision (4)  

**Example 4**: Root (80) => Knowledge Representation (18)  
Agent (4)  
Internet (2)  
Knowledge Acquisition (7)  
Logic Programming (7)  
Machine Learning (7)  
Ontologies (3)  
Semantic Web (3)  
Knowledge Management, Text Mining (3)  

**Example 5**: Root (80) => Knowledge Representation (18) => Artificial Intelligence (14)  
**Parent Topics**:  
Artificial Intelligence (39)  
Knowledge Representation (18)  
**Sub Topics**:  
Agent Theory (3)  
Agent (3)  
Belief Revision (7)  
Causal Reasoning (4)  
Knowledge Acquisition (6)  
Knowledge Discovery (3)  
Logics (7)  
Machine Learning (5)  
Multi-agent systems (3)  
Nonmonotonic reasoning (7)  
Ontologies (2)  
Robotics (2)  
Theory Revision (4)  
Cognitive Modelling, Fuzzy Concepts (2)  

---

**Figure 7.3(b)**: Examples of the browsing structure that evolved.

The term “Knowledge Engineering” is categorised under the term “Artificial Intelligence” (Example 3) and the term “Artificial Intelligence” can also be organised under the “Knowledge Engineering” (Example 5). These structures are based on a lattice which forms a multi-parent relationship as seen in Example 3 and 5. Figure 7.3(c) shows similar examples.
Example 6: Root (80) => Artificial Intelligence (39) => Data Mining (7)

**Parent Topics:**
Artificial Intelligence (39)
Data Mining (17)

**Sub Topics:**
- Database Applications (2)
- Learning (3)
- Machine Learning (5)
- Robotics (2)

Example 7: Root (80) => Data Mining (17)

**Parent Topics:**
Data Mining (17)
Data bases (12)

**Sub Topics:**
- Agent (2)
- Algorithms (2)
- Artificial Intelligence (7)
- Clustering (2)
- Data Structures (2)
- Databases (7)
- Database Applications (6)
- Electronic Commerce (6)
- Information Retrieval (4)
- Machine Learning (6)
- Mobile agent (2)
- XML (4)

Example 8: Root (80) => Data Mining (17) => Databases (12)

**Parent Topics:**
Data Mining (17)
Data bases (12)

**Sub Topics:**
- Computational Geometry (2)
- Database Applications (5)
- Electronic Commerce (4)

Example 9: Root (80) => Databases (12)

**Parent Topics:**
Data Mining (7)
Database Applications (7)
Electronic Commerce (5)
Information Retrieval (5)
Knowledge Discovery (3)
Machine Learning (3)
Semantic Web (4)
Web Services (5)
XML (4)
Knowledge Representation (3)

---

**Figure 7.3(c):** Examples of the browsing structure that evolved.

The main difference between lattice-based browsing and a standard browsing scheme is in the structure of the hierarchy. In the case of a standard browsing scheme, browsing is usually organised in a hierarchical tree structure by locating more general concepts at the top so there is only one path from the root to a given cluster. The lattice allows multiple paths. It would be better to support all practicable structures which reflect all possible inter-relationships within and between objects and their attributes in the system as shown in the examples of Figure 7.3(b) and (c), rather than support only one hierarchy.
For example, the term “Knowledge Representation” is generally categorised under the term “Artificial Intelligence”. However, the structure can also be organised from the “Knowledge Representation” point of view. In other words, the term “Artificial Intelligence” can be organised under the term “Knowledge Representation” as shown Example 5 of Figure 7.3(b).

Of course, in a hierarchical approach, it is also possible to organise one term into a number of clusters. However, relationships between these clusters are specified and maintained by human experts manually in order to keep consistency, but this is not an easy task. This problem will be exacerbated when the size of the knowledge base increases. From this point of view, the concept lattice can have advantages over the hierarchical approach. FCA formulates all possible relationships between terms automatically in accordance with knowledge base updating while maintaining knowledge base consistency.

A more critical advantage of lattice browsing is that it allows one to reach a group of documents via one path, but then rather than going back up the same hierarchy and guessing another starting point, one can go to one of the other parents of the present node as a way of navigating across the domain.

For example, suppose that a user finds “Data Mining” under “Artificial Intelligence”, noticing that there are 7 researchers in this area as shown in Example 6 of Figure 7.3(c). This node has 2 parents and so the lattice view makes it obvious that there are in fact 17 researchers in the School who do research in “Data Mining” as in Example 6 of Figure 7.3(c). If the user goes up to this node, the user then finds that there are nodes with “Data Mining” and “Databases” (see Example 7 and Example 8). The user can then navigate down to these nodes populated by researchers whose more generic interest is databases. These researchers tend to focus on data mining with database applications or database techniques such as association rules, while the AI data-miners tend to use techniques developed in machine learning. There are also other research areas and researchers associated with the term “Data Mining” besides these two groups.
According to our observation on the log of users’ search behaviours, as expected, most users navigated across the lattice - alternatively traversing up different parent concepts and down different child concepts. Generally the users started browsing from a very general term. Then they selected a more specific term of interest that co-occurs with the general term. If a branch centred with the specific term exists in the lattice, they then navigated the branch of the specific term.

For example, suppose that a user starts navigation from the term “Databases” and selects a more specific term “Semantic Web”, a sub-concept of the term “Databases” in the lattice. The user then looks at the search result that displays researchers who are doing research on Databases and the Semantic Web. Then the user usually browses the concept “Semantic Web” as the concept “Databases, Semantic Web” has two parents - “Databases” and “Semantic Web” in the lattice.

From this point of view, the lattice-browsing scheme clearly has advantages over the hierarchical approach where a user simply goes back to the top and starts again. In fact, the hierarchical tree structure, in which each cluster has exactly one parent, is embedded into this lattice structure. Furthermore, as there is a range of views on what an optimal taxonomy might be, use of a lattice approach avoids having to commit to any one taxonomy. The actual preferred usage of terms emerges rather than being prescribed.

7.2.3.2. Survey: Questionnaire on Lattice-based Browsing

This section presents the evaluation data from the on-line Web-based survey which was carried out on lattice-based browsing. Figure 7.4 and 7.5 show the survey questions. The questionnaire was implemented using standard HTML form and JavaScript that let users click on radio buttons, check boxes and enter text and comments into text areas. The implementation style is the same as for the questions on the annotation mechanisms.
Figure 7.4. The first and second questions used in the survey of lattice-based browsing.
Figure 7.5. The third and forth questions used in the survey of lattice-based browsing.

<table>
<thead>
<tr>
<th>Q3</th>
<th>Now that you have tried lattice-based browsing, please give your opinion on searching this sort of a domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(1) Entering search term - boolean query</td>
</tr>
<tr>
<td></td>
<td>(2) Hierarchical browsing - tree structure</td>
</tr>
<tr>
<td></td>
<td>(3) Lattice-based browsing - network structure</td>
</tr>
</tbody>
</table>

| Q3 | Other comments you want to make related to Q3 | 1 2 3 4 5 |

| Q4 | About the system performance and user interface | 1 2 3 4 5 |
|----|---------------------------------------------------------------------------------------------------|
|    | (1) I feel the system is fast enough | too slow | too fast |
|    | (2) I feel the user interface is ok | bad | good |
|    | (3) The help functions are adequate | bad | good |

| Q4 | Other comments you want to make related to Q4 | 1 2 3 4 5 |

**Personal Information**

- (1) An academic in information technology
- (2) A current research student in information technology
- (3) A prospective research student
- (4) An industry person
- (5) From CSE or EE at UNSW
- (6) From elsewhere in Engineering at UNSW
- (7) From elsewhere at UNSW
- (8) Other

You do not have to identify yourself. But it may help us with further follow-up if you do.

**Your Name:**

**Your Email:**

Your last comment - any other comments you can make would be greatly appreciated

Many thanks again,

Minye Kim and Paul Compton

[Send Us Your Data]
**Purpose of the survey**

The objective of this survey was to evaluate the efficiency of lattice-based browsing from the users’ point of view. In addition, the survey aimed at revealing user preferences for search methods in domain-specific document retrieval.

**Methods**

The questionnaire was made available when the system was deployed on the School Web site. There were links to the questionnaire on the browsing pages of the system. E-mails were sent to the researchers in the School inviting them use the system and to complete the survey. To obtain feedback from outside users the link to the questionnaire in the browsing pages was highlighted. The data was collected by a cgi program.

The questionnaire contained 16 questions in four parts. The first part of the questionnaire was to identify the purpose of using the system. The second part aimed to investigating retrieval performance of the system. The third part aimed at identifying user preferences for search methods for a specialised domain (Boolean search, hierarchical browsing and lattice-based browsing). The last part was to measure user satisfaction with the system performance and the user interface. Most questions used a five-point Likert scale to measure users’ view and other questions using the check box format to allow multiple answers.

**Results**

There were 40 questionnaires filled in. Table 7.12 shows the respondents information. Most of the respondents were researchers and current or prospective research students at UNSW (i.e., (1) + (2) + (4) + (5) in Table 7.12). Only one respondent was an outside user from industry, but all respondents were affiliated with information technology.

**Table 7.12.** The respondents’ information.

<table>
<thead>
<tr>
<th>Description</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) A current research student in information technology</td>
<td>19</td>
</tr>
<tr>
<td>(2) A prospective research student</td>
<td>2</td>
</tr>
<tr>
<td>(3) An industry person</td>
<td>1</td>
</tr>
<tr>
<td>(4) From CSE or EE at UNSW</td>
<td>17</td>
</tr>
<tr>
<td>(5) From elsewhere at UNSW</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>40</strong></td>
</tr>
</tbody>
</table>
Table 7.13 shows the main purpose for using the system. Thirty-two of the respondents (80%) were looking at the lattice-based browsing mechanism. As well 65% were looking for specific research areas and 55% were browsing the School of Computer Science and Engineering for study, research and collaboration opportunities. Note that the respondents were allowed to choose multiple items for this question.

Table 7.13. The purpose of the use of the system.

<table>
<thead>
<tr>
<th>Q1. Would you describe your reasons for using this system?</th>
<th>disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Looking for a specific research area</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>17</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.5%</td>
<td>10%</td>
<td>17.5%</td>
<td>42.5%</td>
<td>22.5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) Browsing CSE for study, research and collaboration opportunities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>17.5%</td>
<td>17.5%</td>
<td>22.5%</td>
<td>32.5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) Trying to get an overall impression of CSE research capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>14</td>
<td>11</td>
<td>8</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.5%</td>
<td>35%</td>
<td>27.5%</td>
<td>20%</td>
<td>15%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4) Having a look at a lattice-based browsing mechanism</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>18</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0%</td>
<td>10%</td>
<td>10%</td>
<td>45%</td>
<td>35%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5) Other N/A (No Answer)

Table 7.14 summarises the questionnaire results on retrieval performance of the system. Thirty-eight participants (95% of the respondents) replied that they succeeded in finding what they were looking for, whereas two participants (5%) replied that they failed. There were six individuals among the respondents who chose “YES” who also answered sub-question3 “What were the reasons (for failure)”, as shown in Table 7.15. This means that the six respondents (15%) may have found some relevant information, but not all they expected.
Table 7.14. The questionnaire results on retrieval performance.

Q2. Did you find what you were looking for? *(Yes: 38, No: 2)*

<table>
<thead>
<tr>
<th>If Yes</th>
<th>Responses: 38</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. What did you find?</td>
<td>No. of cases</td>
</tr>
<tr>
<td>(1) An individual researcher and his/her research areas</td>
<td>33</td>
</tr>
<tr>
<td>(2) A broader group of researchers</td>
<td>17</td>
</tr>
<tr>
<td>(3) Some interesting cross-disciplinary areas</td>
<td>13</td>
</tr>
<tr>
<td>2. How did you find it?</td>
<td></td>
</tr>
<tr>
<td>(1) I used mainly</td>
<td>search terms</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) Number of steps to get your result</td>
<td>many steps</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>If No</td>
<td>Responses: 8 <em>(Yes: 6, No: 2)</em></td>
</tr>
<tr>
<td>3. What were the reasons?</td>
<td>No. of cases</td>
</tr>
<tr>
<td>(1) I had a pretty thorough search so I think the area is not covered</td>
<td>4</td>
</tr>
<tr>
<td>(2) The keywords available for browsing were not appropriate</td>
<td>3</td>
</tr>
<tr>
<td>(3) The browsing is too unstructured and I got lost</td>
<td>2</td>
</tr>
<tr>
<td>(4) I was unfamiliar with how to use this system</td>
<td>3</td>
</tr>
</tbody>
</table>

Note: 2. (1)$^x = 3.68$ (2)$^x = 3.76$.

Table 7.15. A cross table with respondents and the reasons they failed for retrieval.

<table>
<thead>
<tr>
<th>Find</th>
<th>(1) Not covered</th>
<th>(2) Keywords not appropriate</th>
<th>(3) Browsing unstructured</th>
<th>(4) Unfamiliar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Respondent1</td>
<td>No</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Respondent2</td>
<td>No</td>
<td></td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Respondent3</td>
<td>Yes</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Respondent4</td>
<td>Yes</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Respondent5</td>
<td>Yes</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Respondent6</td>
<td>Yes</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Respondent7</td>
<td>Yes</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Respondent8</td>
<td>Yes</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
If we have a closer look at the reasons for the failure in sub-question3 in Table 7.15, 4 respondents considered the reasons for failure as being “unfamiliar with how to use the system” and “looking for an uncovered research area” (respondent 3, 4, 6 and 8). On the other hand, another 4 respondents (10%) experienced a failure in finding documents due to “inappropriate keywords available for browsing” or “unstructured browsing” (respondent 1, 2, 5 and 7). These results seem to reveal that there is a need to develop a mechanism to refine a concept lattice in a more structured way for browsing. However, as observed above, 95% of the respondents replied that they succeeded in their retrieval. Conclusions therefore can be made that the search performance of the system is reasonably efficient from the users’ point of view.

The majority of the respondents (87%) indicated that they found individual researchers and their research areas as shown in Figure 7.6. Seventeen participants (45%) indicated that they found a broader group of researchers and thirteen respondents (34%) found some interesting cross-disciplinary areas. Note that the respondents were allowed to choose multiple items for this question. These results indicate that the concept lattice not only forms clusters that locate documents at their proper position, but also formulates interesting inter-relationships among the documents and their concepts (for example, a group of researchers and cross-disciplinary areas). This finding seems to suggest that related documents were found by browsing laterally across the lattice and demonstrate the power of lattice based browsing.

![Figure 7.6. The questionnaire results on “What did you find?”](image)

---

78 Thirty-three of 38 who answered “YES” for Q2.
For the first part of sub-question2 (How did you find it? - I used mainly “search term” to “browsing” in Table 7.14), 66% of the respondents indicated that they mainly used browsing (i.e., rated as 4 or 5 on the 5-point scale), whereas 13% mainly used search terms. Another 21% used both. From the log of users’ search activities, we observed that some respondents who used the topic list replied that they used search terms in finding documents.

There were a number of typical patterns of users’ behaviour when they were looking for documents using the system according to the log of users’ search.

(1) Selecting one topic among the topics listed first and browsing the lattice starting from the node found by the selected topic or from the root of the lattice (iteratively and repeatedly) - 47%.

(2) Entering a search term in the Boolean query interface first and browsing the lattice starting from the node found by the search term (iteratively and repeatedly) - 2%.

(3) Combination (1) and (2) (i.e., select one topic or enter a search term alternatively and browsing the lattice) - 51%.

Most instances of search behaviour pattern of (1) occurred during the annotation process when researchers would be checking their own research topics (i.e., viewing the positioning of their topics in the lattice). Most genuine instances of search involved in the combination pattern (3). Searching was an iterative process - formulating a query and browsing the lattice looking for search results, and changing query terms. Users do not enter a single term and only look at its search result.

For the second part of sub-question2 (Number of steps to get your result - “many steps” to “few steps” in Table 7.14), twenty-eight participants (73.5% of the respondents) replied that few steps were taken to obtain a result. Seven participants (18.5%) rated this question as 3. Only 8% replied that many steps were taken to get a result. Thus, it can be said that the number of steps taken to find a search result was reasonable.

Table 7.16 shows the cross distribution of the responses for the used search methods and the number of steps taken to get a result. Most respondents who rated 2, 3, 4 and 5
for the question “(1) I used mainly ‘search terms’ - ‘browsing’”, indicated that they took a few steps when they were looking for documents. These results seem to show that the search methods both Boolean query and lattice-based browsing provided in the system were efficient. Of those who used mostly browsing (i.e., rated as 5), 75% replied that they took a few steps in finding documents. Of those who used mainly browsing (i.e., rated as 4), 82% regarded the steps taken as few. We can therefore conclude that the browsing search performance was quite efficient; however there is no evidence that browsing is more efficient than using search terms (i.e., Boolean query)\textsuperscript{79}.

Table 7.16. Cross-distribution between the used search methods and the number of steps taken.

<table>
<thead>
<tr>
<th>(2) Number of steps</th>
<th>(1) I used mainly</th>
<th>(2) Number of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>search terms</td>
<td>browsing</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>many steps</td>
<td>1</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>few steps</td>
<td>1</td>
<td>2.5%</td>
</tr>
</tbody>
</table>

Table 7.17 presents the questionnaire results on user opinion on search methods for domain-specific document retrieval. Twenty-five of the respondents (65\%) considered Boolean queries and hierarchical browsing as a helpful in searching a specialised domain. For lattice-based browsing, 90\% of the respondents regarded it as helpful. Note that no one rated lattice-based browsing as 1 or 2. The calculated chi-square value for Table 7.17 is statically significant $\chi^2_s = 13.95$ at 4 degrees of freedom\textsuperscript{80}, indicating that there is a relationship between the search terms and helpfulness. As a consequence, it can be said that lattice-based browsing was regarded as a more helpful search method for domain-specific document retrieval than Boolean query and hierarchical browsing.

\textsuperscript{79} Godin et al. (1993), and Carpineto and Romano (1995; 1996b) evaluated search performance by comparing these two methods. Our experiments have not focused on this comparison.

\textsuperscript{80} This is greater than the critical chi-square values at the level of 95 percent confidence (9.488) and 99 percent confidence (13.277). A more detailed chi-square matrix refers to Appendix 2.
Many users (at least 65%) also thought that Boolean queries and hierarchical browsing would be helpful for searching such a domain as shown in Table 7.17. However, it is not clear that whether users needed or wanted to use both methods or whether they were used to searching with search terms.

| Table 7.17. User opinion on search methods for domain-specific document retrieval. |
|----------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| **Q3. Please give your opinion for searching this sort of a domain** | **unhelpful** | **helpful** | **unhelpful** | **helpful** | **unhelpful** | **helpful** |
| (1) Entering search terms | | | | | | |
|  - boolean query | 1 | 2 | 3 | 4 | 5 | 0 | 5 | 9 | 19 | 7 | 0% 12.5% 22.5% 47.5% 17.5% |
| (2) Hierarchical browsing | | | | | | |
|  - tree structure | 1 | 2 | 3 | 4 | 5 | 0 | 1 | 13 | 19 | 7 | 0% 2.5% 32.5% 47.5% 17.5% |
| (3) Lattice-based browsing | | | | | | |
|  - network structure | 1 | 2 | 3 | 4 | 5 | 0 | 0 | 4 | 25 | 11 | 0% 0% 10% 62.5% 27.5% |

Note: (1) $\bar{x} = 3.7$  (2) $\bar{x} = 3.8$  (3) $\bar{x} = 4.18$.

Table 7.18 shows the cross-distribution between lattice-based browsing and hierarchical browsing choices. Of those who selected lattice-based browsing as a helpful search method for searching in a specialised domain (i.e., rated as 4 or 5), 69% indicated that hierarchical browsing would also be helpful. Of those who responded that they mainly used lattice browsing in finding documents (sub-question 2 of Q2 in Table 7.14), 64% considered that hierarchical browsing would also be a helpful search method.

Table 7.19 shows the cross-distribution between lattice-based browsing and Boolean query choices. Of those who selected lattice-based browsing as helpful (i.e., rated as 4 or 5), 67% thought that Boolean query would also be helpful.

Therefore, we conclude that more diverse search interfaces and methods combined with lattice-based browsing are probably appropriate to meet the different preferences.
Table 7.18. Cross-distribution between lattice-based and hierarchical browsing choices.

<table>
<thead>
<tr>
<th>(2) Hierarchical browsing</th>
<th>(3) Lattice-based browsing</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>N (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>helpful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unhelpful</td>
<td>1 0 0% 0 0% 0 0% 0 0% 0 0% 0 0% 0 0% 0 0% 0 0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 0 0% 0 0% 0 0% 0 0% 0 0% 1 9% 1 2.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 0 0% 0 0% 0 0% 3 75% 8 32% 2 18% 13 32.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 0% 0 0% 0 0% 1 25% 17 68% 1 9% 19 47.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 0 0% 0 0% 0 0% 0 0% 0 0% 7 64% 7 17.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.19. Cross-distribution between lattice-based browsing and Boolean query choices.

<table>
<thead>
<tr>
<th>(1) Boolean query</th>
<th>(3) Lattice-based browsing</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>N (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>helpful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unhelpful</td>
<td>1 0 0% 0 0% 0 0% 0 0% 0 0% 0 0% 0 0% 0 0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 0 0% 0 0% 0 0% 1 25% 4 16% 0 0% 5 12.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 0 0% 0 0% 0 0% 1 25% 8 32% 0 0% 9 22.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 0% 0 0% 0 0% 0 0% 13 52% 6 54.5% 19 47.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 0 0% 0 0% 0 0% 2 50% 0 0% 5 45.5% 7 17.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| N(3) 0 0% 0 0% 0 0% 4 10% 25 62.5% 11 27.5% 40 100% |

Table 7.20 shows user opinion on the system performance and the user interface of the system. Many respondents indicated that they felt the system was fast enough. We installed an Apache Server for Windows NT 4.0 on a personal computer which has 64M RAM and a 200MHz Pentium processor\textsuperscript{81} and expected that the system performance would be slow when connected to the Internet. Sometimes the system performance was so slow that arrangements were made to install the system on a high-capacity computer, but surprisingly the results indicate that many respondents were satisfied with the system performance.

\textsuperscript{81} As indicated earlier, the system was developed with Java, JavaScript and Java Servlets (Java CGI) supported by a Web browser of Netscape 4.0 and Explorer 5.0 or higher.
Table 7.20. The questionnaire results on the system performance and the user interface.

<table>
<thead>
<tr>
<th>Q4. About the system performance and user interface</th>
<th>too slow</th>
<th>N/A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>fast enough</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) I feel the system is fast enough</td>
<td>N/A</td>
<td>1</td>
<td>0</td>
<td>12</td>
<td>15</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0%</td>
<td>2.5%</td>
<td>0%</td>
<td>30%</td>
<td>37.5%</td>
<td>30%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) I feel the user interface is ok</td>
<td>bad</td>
<td>N/A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>good</td>
</tr>
<tr>
<td></td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>20%</td>
<td>55%</td>
<td>25%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) The help function are adequate</td>
<td>bad</td>
<td>N/A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>good</td>
</tr>
<tr>
<td></td>
<td>12.5%</td>
<td>0%</td>
<td>2.5%</td>
<td>32.5%</td>
<td>40%</td>
<td>12.5%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: N/A = No Answer, (1) $\bar{x} = 3.93$ (2) $\bar{x} = 4.05$ (3) $\bar{x} = 3.71$.

It was important that interface comments were positive. The user interface of lattice-based browsing under FCA is usually based on the lattice graph itself. However, our focus was on a Web-based interface using a hypertext representation of the links to a node in a lattice, without a graphical display of the overall lattice. Only a single node and only its immediate parents and children are displayed. The children and parents are hypertext links. Even though graphical views of the whole lattice can give interesting perspectives on the whole domain, they are probably of little interest to someone who is only interested in finding a document and there is an advantage in simplicity.

According to a survey on Internet search engine usage (Chen et al. 2000), features that ranked as most important were ease of use, accuracy, reliability and speed, but “ease of use” ranked relatively high compared to other features. We believe that the hypertext representation is easy to use and a fairly natural simplification of a lattice for Web users and the above results appear to support our belief. It seems to be providing the type of interaction that users are comfortable with on the Web.

Regrading the help function, 52.5% of the respondents indicated that the help function was good with another 32.5% giving a neutral response. Only 2.5% replied that the help function was bad. In addition, 5 participants (12.5% of the respondents) did not answer this question. They may have thought that no help function was available. One
respondent suggested that some explanation of what lattice-based browsing is in the opening page would be helpful. Thus, it can be useful to include a page which gives a more detailed explanation of lattice-based browsing (such as what lattice-based browsing is, how it works and so on) so that users can gain the best use of it.

In fact, there were no visible help functions given. Only a brief explanation on lattice-based browsing was given at a link “About the system” in the search pages. Hence, it might be said that only those 5 participants who did not respond (12.5% of the respondents) gave the right answer. However, the respondents who answered this question may have felt no need for help functions. They may have thought that the user interface provided was quite comprehensive and natural to use. Note that no one indicated that the user interface was bad in the second question of Q4 as shown in Table 7.20. Most respondents replied that the user interface was satisfactory.

In summary, the survey respondents indicated that the search performance of the system was efficient. As well the number of steps needed to get a search result was small or at least reasonable. The majority of the respondents replied that they found individual researchers and their research areas. Some indicated that they also found a broader group of researchers and some interesting cross-disciplinary areas. This seems to support the hypothesis that the concept lattice not only formulates clusters which locate documents at their proper position, but also formulates additional inter-relationships among the documents establishing research groups and cross-disciplinary areas.

Many respondents indicated that they were satisfied with the system performance as well as the user interface. It seems that the user interface was fairly comprehensible and natural to use for Web users, because many respondents considered that the help questions was adequate, even though there were no concrete help functions given.

Lattice-based browsing was considered as a more helpful search method for domain specific document retrieval than Boolean query or hierarchical browsing. Nevertheless, many respondents considered not only lattice-based browsing, but also Boolean query and hierarchical browsing would be helpful for searching such a domain. This seems to
suggest that more flexible combination techniques are desirable to meet the different needs of users.

On the other hand, there seems to be a requirement to refine a concept lattice in a more structured way with more appropriate terms as some of the respondents indicated that they have experienced failure in finding documents due to inappropriate keywords available for browsing or unstructured browsing. However, this does not negate the statistical conclusion that lattice based browsing is more helpful than Boolean queries and hierarchical browsing. Our conclusion is that lattice based browsing is a step forward but does not fully overcome all the problems in searching for documents.

7.3. Chapter Summary

This chapter presented the experimental results on the proposed system for the domain of research interests in the School of Computer Science and Engineering, UNSW. The annotation mechanisms available to annotators provided a good level of assistance. The most interesting result suggested that although an established external taxonomy could be useful in suggesting annotation terms, small communities appeared to have little interest in adhering to “standard” hierarchical structures. This result confirms one of our motivations for this thesis – it is advantageous to allow users in small communities very flexible access to established ontological resources, so the most appropriate use for the community can emerge.

A browsing structure that evolves in an ad hoc fashion provided good efficiency in search performance. In addition, lattice-based browsing was considered as a more helpful method than Boolean query or hierarchical browsing for searching a specialised domain. Moreover, many users were satisfied with the system performance as well as the user interface of browsing. The experimental results seem to support the hypothesis of the power of lattice-based browsing over the hierarchical approach – the lattice structure showed that the pathway traversed found only a small number of documents, but there were other related documents that came out of other research approaches. It also showed users alternative pathways to a node, which the users had not yet navigated, which might lead to documents of interest.
Chapter 8
Discussion and Conclusion

This chapter presents a summary of the thesis. We then conclude with a discussion of possible future directions of the research presented in this thesis.

8.1. Motivation

Most work on document management and retrieval intended for Web-based documents, focuses on either improved search engines or ontology development. The assumptions with the ontology approach seem to be that since communities do communicate, there must be consensus about terms; so the main thing that needs to be done is to identify and formalise this consensus. This will then result in a standard ontology, and anyone wishing to communicate in a domain, will be keen to use the standard ontology and reap the benefits that any documents that use the standard ontology will be much more readily retrieved and used by others.

We have no dispute with this, except the lack of focus on how the consensus represented by an ontology emerges and evolves. For example, the classic paper by Shaw (1988) on use of terms in geology showed, that left to their own devices, geologists described geology in quite different terms and that they disagreed and misunderstood the sets of terms they had independently developed. However, she concluded that despite this apparent confusion, these geologists had little trouble in working together and understanding each other when working together. This suggests that attempting to get a consensus that everyone agrees with and then works within will be difficult. In this regard, it appears that the working group on an upper level ontology, has effectively broken up in disarray, unable to agree on an ontology (Gangemi et al. 2002). In fact, some ontology researchers like Gangemi et al. see their
goal as developing formalisms to assist in designing proper ontologies – but that these will be disposable, and their only value will be in how much they are used.

Our approach has been rather to look for tools and techniques by which a de-facto consensus might emerge and might evolve further. The main application of this is for small groups and communities. Any such tools would need excellent browsing, particularly to find related but unexpected concepts and tools to assist users to re-use terms used by others, but not to constrain them.

The system developed to meet these goals uses Formal Concept Analysis (FCA) to support flexible browsing and has a number of mechanisms to encourage others to re-use terms. In particular, it encourages users to select terms from external ontologies, where such exist.

8.2. Summary of Results

We implemented this system and carried out an evaluation in using it to assist prospective students and others find research supervisors and collaborators in the School of Computer Science and Engineering, UNSW. We logged users’ actions in browsing and annotating, and some users filled in a Web questionnaire.

8.2.1. Annotation Mechanisms

A number of knowledge acquisition techniques and tools were developed to suggest possible annotations. The survey results indicate that the various annotation tools assisted the users in defining their research topics so that the lattice-based browsing structure that evolved in an ad hoc fashion was organised into a reasonable consensus with good efficiency in retrieval performance. In particular, it should be noted that there was no training or help provided for the users. The aim was to have a self explanatory system. However, we do not have results that demonstrate that the users were more effective in annotating their pages with these mechanisms than without.
8.2.2. Lattice-based Browsing

It was clear from the results that there is an advantage in lattice-based browsing over a hierarchical approach. If one fails to find the appropriate document, one can ascend to the top of the lattice by another pathway. In other words, lattice browsing allows one to reach a group of documents via one path, but then rather than going back up the hierarchy and guessing another starting point, one can go to one of the other parents of the present node as a way of navigating across the domain. The critical problem with hierarchical browsing is that if a user does not find the required document the user will not be sure what to do next - the user has already made his/her best guesses at various decision points. Lattice-based browsing shows the user alternative pathways to a node, which the user has not yet navigated and which may lead to documents of interest.

The survey results and examples shown support this hypothesis and demonstrate that lattice browsing can help the user find both what they are looking for and also interesting related documents.

8.2.3. Web-based System

Another emphasis of our approach was a Web-based system using a hypertext representation of the links to a node in a lattice without a graphical display of the overall lattice. We focused on simplicity and familiarity for Web users. The survey results indicate that the Web implementation we used provides a fairly easy environment for users who are familiar with the Web. In contrast, although graphical views of the whole lattice may give an interesting perspective on the overall domain, they are probably of little use to someone who is interested in finding a document.

8.2.4. Imported Ontologies

The one area where there was a strong result from the way users behaved was with using imported ontologies. We were able to demonstrate that users did use external ontologies, but did so very selectively. The results suggest that the external ontologies are of value as a resource, but in small communities and specialised domains, people prefer to pick and choose what is of value from ontologies. It seems likely that within a
small community, even a quite diverse community, selective use will be made of a more
global ontology and this usage pattern can itself become a useful ontology for other
groups.

It should be noted that in this particular application one would have expected the
taxonomies used to have a reasonable match with the group annotating documents. So
the usage of terms probably says something about the relevance of these terms to the
task of identifying the research interests of an individual. For example, the term
“knowledge engineering” although seemingly a useful concept, probably gives little
idea to prospective students and collaborators of the particular style of research carried
out, since under this term there are some very different areas of research. Hence,
although it was suggested from the external ontologies, it was little used. On the other
hand, “knowledge representation” covers a much more coherent style of research.

8.3. Expectations for Other Domains

A major issue is whether the experience we have had in this domain will apply to other
domains; i.e. will the complexity of lattice become too great in other domains.

In a lattice structure, all possible document subsets can produce an exponential number
of lattice nodes. However, Godin et al. (1986) examined the worst case time complexity
of a lattice structure linearly bounded with the number of documents when the number
of terms for each document has an upper bound, which is usually the case in practical
applications; i.e., \(|H^*| \leq 2^k n\) where \(|H^*|\) is the number of all formal concepts, \(n\) is the
number of documents and \(k\) is the mean number of terms per document. This means that
\(|H^*| / n\) is much smaller than the upper bound \(2^k\). In fact, their experiments in several
domains showed that in every application, \(|H^*| / n \leq k\). For example, in one of the
experiments with 3042 documents and an average 11.1 terms per document, 23471
lattice nodes were produced. This means that the average number of lattice nodes per
document 7.7 (23471/3042) is much less than the upper bound 2^{11.1}. In our experiment,
a document is associated with 5.9 concept nodes (471/80) with an average 7.97 terms
for a homepage. Godin et al. (1986) also indicated that average values for search time with the hierarchical method (3.90 min) are slightly better than the lattice method (3.95 min). But the difference is not significant. Hence, we anticipate that the complexity of the lattice structure will not be a significant problem. In addition, one can surmise that since single inheritance hierarchies work reasonably well, without massive repetition of concepts in different part of the hierarchies, a lattice is unlikely to have a very high degree of interconnection. That is, although the complexity is potentially great, there is no evidence that the world is such that this will occur.

However, in the limit, it is possible that the lattice will become too complex. Nevertheless, we do not expect this to occur since, as we believe that any co-operative effort in lattice building will converge towards a system that is reasonably useable. Since the annotator is highly likely to use the lattice first and is also prompted by the terms that are used, they are likely to converge on a reasonable size. There is a tension for users between adding a document with sufficient annotation terms so that it is distinguished from all other documents, but on the other hand leaving the lattice sufficiently useable so that the document can be found.

The use of the ontology editor and the role of some sort of lattice manager (knowledge engineer) may also be significant. If the initial concepts added are too fine grained, then the ontology editor is likely to be used to produce conceptual scales to group the refined concepts into a broader concept. A further extension would be that the manager might hide very refined concepts, however we have not yet included this facility.

As well, there are further possibilities of developing separate sub-lattices if the community takes on a domain that is too big for a single lattice. Again this could be achieved using conceptual scales.

8.4. Future Work

Although we believe FCA is a useful way of supporting the flexible open management of documents, there are a number of areas of further development that need to be explored.
8.4.1. Ontologies

The most interesting result from our study was in the use of the imported taxonomies. However, there are a number of areas requiring further research for improvement of our approach regarding ontologies.

**Importing Standard Format Ontologies:** First of all, it will be essential to deal with Web-based ontology representation languages such as XML, OIL, DAML or DAML+OIL instead of the proprietary text formats used currently. Annotation mechanisms which commit to ontologies, mostly use one of these ontology representation languages. The main aim of using these languages is to facilitate the sharing of information between communities (or agents) as well as individuals within the communities. A Web Ontology Working Group has also been organised to construct a standard Ontology Web Language (OWL) for the Semantic Web.

Note that the FCA community has founded the Tockit project\(^{82}\) “to write a framework for Conceptual Knowledge Processing” (http://tockit.sourceforge.net/tockit/index.html, 2003). The aims of this project include defining a XML standard for Formal Concept Analysis and ontology guided document retrieval.

We will move to a system whereby the user can simply specify any URL from where an ontology in a standard format can be imported and used in the system.

**Importing Conceptual Scales:** Another area requiring further attention is related to importing the ontological structure of a domain into conceptual scales. We have adapted conceptual scaling of FCA to scale up the browsing structure of the system with ontological information where readily available such as author, person, academic position, research group and so on. These correspond to the type of more structured ontology information used in the system such as KA\(^2\). We included such information for interest in this study. However, ideally we would desire conceptual scales from an imported ontology. The use of these scales could be automated if the document was appropriately marked up according to the ontology. This would give us a system that

---

was flexible and open, but also had the type of ontological commitment represented by
the KA\textsuperscript{2} initiative. It will be interesting to examine the trade-offs in allowing such
requirements to emerge rather than anticipating them and also the relative costs in
marking up documents rather than providing information to a server.

**Ontology Editing:** We have implemented a tool which allows a knowledge engineer (or
user) to identify abbreviations, synonyms or groupings. The group hierarchy is then
used for conceptual scales. The tool is a fairly simple and standard editor. But it allows
only a single inheritance hierarchy in each grouping, so that an extension to the tool
may be required to be able to handle more complex ontologies. Note that there are a
number of well-established ontology editors in the literature such as Protégé\textsuperscript{83}, OilEd\textsuperscript{84}
and OntoEdit\textsuperscript{85}.

As a possible alternative for handling more complex ontologies, a user can be allowed
to group ontological attributes (i.e., groupings) and also to build a hierarchy of groups
of groupings. The groupings can be named. This would be a hierarchical representation
of an “ontology” similarly to how a browsing ontology is presented in the KA\textsuperscript{2}
initiative. These “ontologies” would be constructed on the fly, but stored for future use
if required. The user would be free to select any one of these ontologies to interact with
the system, and use this interaction to move to a particular sub-lattice. In this area it
may be possible to use some form of machine learning to select likely candidate nodes
for grouping together.

**Multiple Ontology View:** We have imported a number of taxonomies (i.e., ACM,
ASIS&T and Open Directory Project) from commonly available Web sites to suggest
possible annotations. Another aim of importing ontologies is to give a different lattice
view based on one of these taxonomies. This is to assist people who would be expected
to have a more superficial knowledge of the terms used for document annotation based
on a certain taxonomy. As ontology representation standards become better established,

\textsuperscript{83} http://protege.stanford.edu/ (2003).
\textsuperscript{84} http://oiled.man.ac.uk/ (2003).
importing taxonomies and using them to give a different lattice view should only require entering a URL. As well, different users may develop different ontological views using conceptual scales. These would also be able to be selected by other users.

**Exporting Ontologies:** The various ontological views that are developed would also be exported using standard ontology formats. The current system keeps the annotation separated from the actual documents. If desired, documents can be copied with their annotation added according to standard mark-up languages and exported for use with other software.

### 8.4.2. Annotation Support

The system supports a number of annotation tools to assist users to easily annotate their documents and to be able to reuse terms used by others, and those imported from taxonomies. However, it would also be useful to use machine learning or natural language processing techniques to identify “*key terms and phrases*” and/or “*ontological attributes*” from the annotated document. These suggestions could be integrated with the tools that the current system uses. Note that some studies (Aussenac-Gilles et al. 2000; Maedche and Staab 2000; Handschuh et al. 2002) have investigated automated or semi-automated ways of discovering the appropriate ontology for a document.

Another issue that we anticipate is the handling of documents which consist of several sections that are about different topics. In the current system, each document is represented as a bag of words (i.e., a set of keywords) which are simply summed up across the different sections.

### 8.4.3. Integration with Other Techniques

As described earlier, we have integrated a number of information retrieval techniques into lattice browsing. Firstly, a Boolean query interface is combined with the FCA browsing interface with some normalisation techniques such as eliminating stopwords, stemming and expanding user queries based on synonyms and abbreviations. Secondly,
a textword search is supported when the system fails to get a result from the lattice and is used to show a sub-lattice. However, it is also likely that there are a number of areas where our approach may be integrated with general Web search engines and their techniques.

A simple change would be to replace the textword search engine with a standard engine such as Google. However, this may require all the referred documents to be copied to a single site. One might then be able to set up mechanisms to move seamlessly from a search of annotated documents to a search of the whole Web.

**8.4.4. Security and Extension**

Anyone can access and browse the lattice to find information within the system. However, for annotations, only staff and research students of the School of Computer Science and Engineering, UNSW can annotate documents for research topics. The only documents the system provides access to, are the home pages of these staff and students. We use the local Unix account at the School to authenticate users for the annotation. This also provides a default home page address of the users.

However, different applications (i.e., in the domain of the Banff Knowledge Acquisition Proceedings papers) will require different types of authentication. There is a need to set up a study in which there are no controls at all – anyone with access to the Web can set-up and change annotations to any page on the WWW. We imagine this may be of considerable interest to people using open discussion lists in various communities.

Further work would be related to the extension of the approach across other related groups which potential collaborators belong to, or across communities which share the same domain knowledge beyond a small department. Integration strategies of browsing across the disparate communities with appropriate security will be an interesting issue.
8.5. Conclusion

We have presented a Web-based incremental document management and retrieval system for small communities in specialised domains based on the concept lattice of Formal Concept Analysis. The incremental approach we have used has a similar motivation to both Ripple-Down Rules and Repertory Grids. FCA facilitates browsing and users adding documents seem to enjoy seeing how their documents fit into the lattice and making sure they are appropriately positioned. The users have a much more flexible role, adding terms as the need arises.

The browsing structure that evolves in an ad hoc fashion evolved into a reasonable consensus and provided good efficiency in retrieval performance. In general, lattice-based browsing was considered by users as a more helpful search method than Boolean queries and hierarchical browsing for searching a specialised domain. The experimental results also supported the hypothesis that lattice-based browsing is more powerful than a hierarchical approach. Users were satisfied with the system performance and the Web interface for lattice-based browsing. An interesting result suggested that although an established external taxonomy could be useful in proposing annotation terms, small communities appeared to have little interest in adhering to standard taxonomies and users appeared to be very selective in their use of terms proposed.

However, this does not mean that FCA solves all of the problems in managing and retrieving documents for specialised domains. There are a number of areas requiring further research particularly relating to ontologies and automated annotation support.

We conclude that the concept lattice of Formal Concept Analysis, supported by annotation tools is a useful way of supporting the flexible open management of documents and retrieval required by individuals and small communities in specialised domains. It seems likely that this approach can be readily integrated with other developments such as further improvements in search engines and the use of semantically marked-up documents. This would result in a seamless linking of general search to ad hoc ontologies through to established standard ontologies.
We also suggest that in the near term as standards for representing ontologies take hold, these small community systems will play an important role in helping to develop standard ontologies. However, rather than being locked into conforming to the standard, users will be free to use all, small fragments, or none of the ontology as best suits their purpose; that is, these communities will be able to very flexibly import ontologies and make selective use of ontology resources. Their selective use and the extra terms they add will provide useful feedback on how the external ontologies could be evolved. A new ontology will emerge as the result, and this itself may become a new standard ontology.
### Appendix

#### A.1 Retrieval Performance on the Queries in Table 7.11

| Terms                        | Number of researcher home pages retrieved |  |  |  |  |  |  |  |
|------------------------------|------------------------------------------|---|---|---|---|---|---|
|                              | Lattice only | ACM Taxonomy | ASIS&T Taxonomy | Open Directory Project | UNSW Taxonomy |  |  |
|                              | Retrieved    | Precision    | Retrieved       | Precision              | Retrieved     | Precision | Retrieved | Precision |
| Artificial intelligence      | 39           | 1            | 50              | 0.78                   | 45           | 0.87       | 56        | 0.70       | 58        | 0.67 |
| Knowledge-engineering        | 3            | 1            | -               | -                      | 32           | 0.09       | -         | 30         | 0.10 |
| Knowledge-representation     | 18           | 1            | 18              | 1.00                   | 20           | 0.90       | 22        | 0.82       | 22        | 0.82 |
| Knowledge-management         | 4            | 1            | -               | -                      | 25           | 0.16       | 25        | 0.16 |
| Knowledge-discovery          | 7            | 1            | -               | -                      | 24           | 0.29       | 24        | 0.29 |
| Machine-learning             | 24           | 1            | -               | 24                     | 1.00         | 28         | 0.86      | 28         | 0.86 |
| Learning                     | 5            | 1            | 21              | 0.24                   | -            | -         | -         | -         |
| Information-processing       | 1            | 1            | -               | 11                     | 0.09         | -         | -         |
| Information-retrieval        | 10           | 1            | 11              | 0.91                   | 10           | 1.00       | 11        | 0.91       | 11        | 0.91 |
| Internet                     | 4            | 1            | 4               | 1.00                   | 6            | 0.67       | 6         | 0.67       | 6         | 0.67 |
| Databases                    | 11           | 1            | 11              | 1.00                   | 12           | 0.92       | 22        | 0.50       | 13        | 0.85 |
| Computer-programming         | 1            | 1            | 11              | 0.09                   | 9            | 0.11       | 1         | 1.00       | 11        | 0.09 |
| Programming-languages        | 4            | 1            | 4               | 1.00                   | 4            | 1.00       | 4         | 1.00       | 4         | 1.00 |

**Average Precision**

An average decrease = $1 - (0.75 + 0.58 + 0.69 + 0.58) / 4 = 0.35$. 

185
### A.2 Chi-Square ($\chi^2$) Matrix for Table 7.17

<table>
<thead>
<tr>
<th>Search methods</th>
<th>Boolean query</th>
<th>Hierarchical browsing</th>
<th>Lattice-based browsing</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f_o$</td>
<td>%</td>
<td>$f_e$</td>
<td>%</td>
</tr>
<tr>
<td>1-2</td>
<td>5</td>
<td>12.5</td>
<td>2.0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>22.5</td>
<td>8.7</td>
<td>13</td>
</tr>
<tr>
<td>4-5</td>
<td>26</td>
<td>65.0</td>
<td>29.3</td>
<td>26</td>
</tr>
<tr>
<td>Total</td>
<td>40</td>
<td>100.0</td>
<td>40.0</td>
<td>40</td>
</tr>
</tbody>
</table>

Note that the chi-square equation is $\chi^2 = \sum \frac{(f_o - f_e)^2}{f_e}$ where $f_o$ is the frequency obtained and $f_e$ is the frequency expected in each cell under the assumption of no difference.

### A.3 Critical Values of the Chi-Square Distribution used in Chapter 7

<table>
<thead>
<tr>
<th>df</th>
<th>$\chi^2_{0.05}$</th>
<th>$\chi^2_{0.01}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.841</td>
<td>6.635</td>
</tr>
<tr>
<td>2</td>
<td>5.991</td>
<td>9.210</td>
</tr>
<tr>
<td>3</td>
<td>7.815</td>
<td>11.345</td>
</tr>
<tr>
<td>4</td>
<td>9.488</td>
<td>13.277</td>
</tr>
<tr>
<td>5</td>
<td>11.070</td>
<td>15.086</td>
</tr>
<tr>
<td>6</td>
<td>12.592</td>
<td>16.812</td>
</tr>
<tr>
<td>7</td>
<td>14.067</td>
<td>18.475</td>
</tr>
<tr>
<td>8</td>
<td>15.507</td>
<td>20.090</td>
</tr>
<tr>
<td>9</td>
<td>16.919</td>
<td>21.666</td>
</tr>
<tr>
<td>10</td>
<td>18.307</td>
<td>23.209</td>
</tr>
<tr>
<td>11</td>
<td>19.675</td>
<td>24.725</td>
</tr>
<tr>
<td>12</td>
<td>21.026</td>
<td>26.217</td>
</tr>
<tr>
<td>13</td>
<td>22.362</td>
<td>27.688</td>
</tr>
<tr>
<td>14</td>
<td>23.685</td>
<td>29.141</td>
</tr>
<tr>
<td>15</td>
<td>24.996</td>
<td>30.578</td>
</tr>
<tr>
<td>16</td>
<td>26.587</td>
<td>31.000</td>
</tr>
<tr>
<td>17</td>
<td>27.587</td>
<td>33.409</td>
</tr>
<tr>
<td>18</td>
<td>28.869</td>
<td>34.805</td>
</tr>
<tr>
<td>19</td>
<td>30.144</td>
<td>36.191</td>
</tr>
<tr>
<td>20</td>
<td>31.410</td>
<td>37.566</td>
</tr>
</tbody>
</table>
Bibliography


