UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE

P.O. BOX 2208, HERAKLION, CRETE,
GR-714 09 GREECE

On Belief Change and Ontology Evolution

Giorgos Flouris

fgeo@csd.uoc.gr

Doctoral Dissertation

Heraklion, February 2006

Copyright © by Giorgos Flouris, 2006
All Rights Reserved

UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE

On Belief Change and Ontology Evolution

Dissertation submitted by
Giorgos Flouris
in partial fulfillment of the requirements for
the PhD degree in Computer Science

Author:

Giorgos Flouris, University of Crete

Examination Committee:

Dimitris Plexousakis, Associate Professor, University of Crete, Supervisor

Grigoris Antoniou, Professor, University of Crete

Nicolas Spyratos, Professor, University of Paris-South

Panos Constantopoulos, Professor, Athens University of Economics and Business

Vassilis Christophides, Associate Professor, University of Crete

Manolis Koubarakis, Associate Professor, National and Kapodistrian University of Athens

Pavlos Peppas, Associate Professor, University of Patras

Approved by:

Dimitris Plexousakis
Chairman of Graduate Studies
Heraklion, February 2006

Abstract

One of the crucial tasks to be performed towards the realization of the vision of the Semantic Web is the encoding of human knowledge in special structures (ontologies), using formal representation languages. Simply creating an ontology is not enough though; knowledge needs to be updated as well. There are several reasons why knowledge in an ontology would need to change: changes could be initiated because of a change in the world being modeled; the users' needs may change, requiring a different conceptualization; knowledge previously unknown, classified or otherwise unavailable may become known; or a design flaw may have been noticed in the original conceptualization.

In all these cases, the ontological representation would need to be modified so as to form a more accurate conceptualization of the domain. In this dissertation, we will use the generic term ontology change to refer to the process of modifying an ontology in response to any given need. This generic term engulfs several different, but closely related research fields, including ontology evolution, alignment, merging, mapping and others.

The first part of this dissertation deals with all these disciplines, performing a shallow, yet broad review of recent work in those areas and proposing a certain terminology in a field plagued by the use of underspecified and confusing terms. This will allow us to draw a fine line between these closely related and often overlapping research efforts.

The second part of this dissertation focuses on ontology evolution, which refers to the process of modifying an ontology in response to a change in the domain or its conceptualization. We will argue that current research on ontology evolution is misguided and leads to several problems, mostly related to the heavy dependency of the process on human participation and the lack of adequate formalizations. The key idea of our proposal towards resolving these deficiencies is the use of intuitions from the closely related field of belief change, which deals with the updating of knowledge stored in Knowledge Bases.

We will study the connections between the two fields and propose a formalization for ontology evolution that facilitates the migration of belief change techniques, tools, ideas, results and intuitions to the problem of ontology evolution. The solutions proposed in the belief change literature generally have all the properties desirable for an ontology evolution algorithm (automatic determination and implementation of changes, formal character and well-studied theoretical properties), so they could be of use in the ontology evolution field. This way, ontology evolution researchers will not have to "reinvent the wheel" for problems whose counterparts have already been studied in the belief change literature and will be protected from potential pitfalls and dead ends.

It is obvious that exhaustively pursuing the migration of belief change techniques to the ontology evolution field is a long-term goal, not achievable within the framework and

limitations of a PhD dissertation. For this reason, this part of our work will only provide abstract ideas towards this direction. Our aim at this point will be merely to provide the necessary theoretical foundations and connections upon which future research could be based, hopefully leading to more concrete solutions to the problem of ontology evolution.

Following this abstract study, our research will focus on a certain belief change theory, namely the AGM theory, which is the most influential approach in the field of belief change. This study constitutes the third part of this thesis and its aim is to provide an exemplary application of the general research path we are proposing, as well as to show its feasibility.

Our study of the AGM theory will be initiated by an attempt to generalize it so as to be applicable to logics that are used for ontological representation. The AGM theory, like most belief change approaches, focuses on classical logic, disallowing, among others, the main formalisms used in ontologies, like Description Logics (DLs) and the Web Ontology Language (OWL). Our generalization will cover a vast class of monotonic logics and show that the AGM theory is actually applicable to a certain class of logics much wider than the original class studied by AGM. Our study will also uncover interesting theoretical properties of the AGM theory, as well as its connection with other formalisms, like the foundational model and lattice theory.

In the fourth part of this dissertation, our focus will be further narrowed so as to refer to the main ontology representation formalisms, namely DLs and OWL. We will provide specialized conditions under which members of these families of logics are compatible (or incompatible) with the AGM theory, as well as several heuristics and results allowing one to determine this compatibility (or incompatibility) for logics not covered by this work. The main results of this part of our study is that certain DLs, as well as OWL, are not compatible with the AGM theory, while in other DLs the AGM theory can be successfully applied for the development of ontology evolution algorithms; for the latter category of DLs, an algorithm (operator) performing contraction that is compatible with the AGM theory is described.

Supervisor:
Dimitris Plexousakis,
Associate Professor of Computer Science
University of Crete

Περίληψη

Μία από τις πιο σημαντικές εργασίες προς την υλοποίηση του Σημασιολογικού Ιστού (Semantic Web) είναι η κωδικοποίηση της ανθρώπινης γνώσης σε ειδικές δομές (οντολογίες), χρησιμοποιώντας τυπικές γλώσσες αναπαράστασης. Η απλή δημιουργία μιας οντολογίας δεν είναι αρκετή όμως, διότι η γνώση πρέπει επίσης να ενημερώνεται. Υπάρχουν διάφοροι λόγοι για αυτό: αλλαγές θα μπορούσαν να προκληθούν από μία αλλαγή στον κόσμο υπό μοντελοποίηση, από κάποια αλλαγή στις ανάγκες των χρηστών, από την απόκτηση πρόσβασης σε γνώση προηγουμένως άγνωστη, απόρρητη ή απλώς μη διαθέσιμη, ή από τον εντοπισμό κάποιου προβλήματος στον αρχικό σχεδιασμό της οντολογίας.

Σε όλες αυτές τις περιπτώσεις, η αναπαράσταση της οντολογίας θα πρέπει να τροποποιηθεί κατάλληλα ώστε να μοντελοποιεί ακριβέστερα την περιοχή ενδιαφέροντος. Σε αυτήν την διατριβή, θα χρησιμοποιήσουμε το γενικό όρο «αλλαγή οντολογιών» (ontology change) αναφερόμενοι στη γενικότερη διαδικασία αλλαγής μιας οντολογίας εξαιτίας κάποιας δεδομένης ανάγκης. Αυτός ο γενικός όρος περιλαμβάνει αρκετούς διαφορετικούς, αλλά πολύ σχετικούς μεταξύ τους ερευνητικούς τομείς, όπως η εξέλιξη (evolution), ευθυγράμμιση (alignment), συγχώνευση (merging) και απεικόνιση (mapping) οντολογιών.

Το πρώτο μέρος αυτής της διατριβής εξετάζει όλους αυτούς τους τομείς, εκτελώντας μία επιφανειακή, αλλά ευρεία μελέτη των πρόσφατων εργασιών στις περιοχές αυτές. Επίσης, προτείνεται κάποια συγκεκριμένη ορολογία σε έναν τομέα ο οποίος μαστίζεται από την χρήση όχι ξεκάθαρων και μη επαρκώς ορισμένων όρων, οι οποίοι συχνά προκαλούν σύγχυση. Αυτή η μελέτη θα επιτρέψει τον διαχωρισμό αυτών των πολύ σχετικών και συχνά επικαλυπτόμενων ερευνητικών προσπαθειών.

Στο δεύτερο μέρος της εργασίας θα εστιάσουμε στο πρόβλημα της εξέλιξης οντολογιών (ontology evolution), που ασχολείται με την αλλαγή μιας οντολογίας εξαιτίας κάποιας αλλαγής στον κόσμο υπό μοντελοποίηση ή στην ίδια την μοντελοποίηση. Θα υποστηρίξουμε ότι οι τρέχουσες σχετικές ερευνητικές προσπάθειες οδηγούν σε διάφορα προβλήματα, σχετιζόμενα κυρίως με την εξάρτηση της διαδικασίας από την συμμετοχή του χρήστη και με την έλλειψη επαρκούς τυποποίησής της. Η πρότασή μας σχετικά με την επίλυση αυτών των προβλημάτων βασίζεται στην χρήση ιδεών από τον σχετικό ερευνητικό τομέα της αναθεώρησης πεποιθήσεων (belief change), ο οποίος μελετάει το πρόβλημα της ενημέρωσης πληροφορίας αποθηκευμένης σε βάσεις γνώσεων (Knowledge Bases).

Θα μελετήσουμε την σχέση μεταξύ των δύο τομέων και θα προτείνουμε ένα τυπικό πλαίσιο εργασίας για το πρόβλημα της εξέλιξης οντολογιών το οποίο θα διευκολύνει τη χρήση τεχνικών, εργαλείων, ιδεών και αποτελεσμάτων από το πεδίο της αναθεώρησης πεποιθήσεων στο πρόβλημα της εξέλιξης οντολογιών. Οι λύσεις που προτείνονται στη βιβλιογραφία αναθεώρησης πεποιθήσεων έχουν γενικά όλες τις επιθυμητές ιδιότητες ενός

αλγορίθμου εξέλιξης οντολογιών, όπως ο αυτόματος προσδιορισμός και εφαρμογή των αλλαγών, ο τυπικός χαρακτήρας των προσεγγίσεων, καθώς και η σε βάθος μελέτη των θεωρητικών τους ιδιοτήτων. Επομένως, τέτοιες τεχνικές θα μπορούσαν να είναι χρήσιμες και στο πεδίο της εξέλιξης οντολογιών. Έτσι, ερευνητές που ασχολούνται με την εξέλιξη οντολογιών δεν θα χρειάζεται «να εφεύρουν τον τροχό» προκειμένου να αντιμετωπίσουν προβλήματα των οποίων αντίστοιχα έχουν ήδη μελετηθεί στη βιβλιογραφία αναθεώρησης πεποιθήσεων και θα προστατεύονται από πιθανές ερευνητικές «παγίδες» ή αδιέξοδα.

Είναι προφανές ότι η εξαντλητική μελέτη της χρήσης τεχνικών από το πεδίο της αναθεώρησης πεποιθήσεων στο πεδίο της εξέλιξης οντολογιών είναι ένας μακροπρόθεσμος στόχος, μη επιτεύξιμος μέσα στα πλαίσια και τους περιορισμούς μιας διδακτορικής διατριβής. Για τον λόγο αυτό, η παραπάνω μελέτη παρέχει μόνο κάποιες αφηρημένες ιδέες προς την κατεύθυνση αυτή. Στόχος μας στο σημείο αυτό είναι η δημιουργία των απαραίτητων θεωρητικών βάσεων πάνω στις οποίες θα μπορούσε να βασιστεί η μελλοντική έρευνα ώστε να οδηγήσει σε πιο συγκεκριμένες λύσεις στο πρόβλημα της εξέλιξης οντολογιών.

Στην συνέχεια, η έρευνά μας θα εστιάσει σε μια συγκεκριμένη θεωρία αναθεώρησης πεποιθήσεων, την θεωρία των AGM, η οποία είναι η σημαντικότερη προσέγγιση στον τομέα αυτόν. Αυτή η μελέτη αποτελεί το τρίτο μέρος της διατριβής και στόχος της είναι να προβάλει μια πιθανή εφαρμογή της γενικής ερευνητικής πορείας που προτείνουμε, καθώς επίσης και να αποδείξει την δυνατότητα πραγματοποίησής της.

Η μελέτη της θεωρίας των AGM θα αρχίσει με μια προσπάθεια γενίκευσής της ώστε να μπορεί να εφαρμοστεί σε λογικές που χρησιμοποιούνται για αναπαράσταση οντολογιών. Η θεωρία των AGM, όπως και οι περισσότερες προσεγγίσεις αναθεώρησης πεποιθήσεων, εστιάζει στην κλασσική λογική, με συνέπεια να μην είναι εφαρμόσιμη στους κύριους φορμαλισμούς αναπαράστασης οντολογιών, όπως οι Περιγραφικές Λογικές (Description Logics – DLs) και η γλώσσα OWL (Web Ontology Language). Η παραπάνω γενίκευση καλύπτει μια ευρεία κατηγορία λογικών και αποδεικνύει ότι η θεωρία των AGM μπορεί να εφαρμοστεί σε μια κλάση λογικών ευρύτερη αυτής η οποία είχε αρχικά μελετηθεί από τους AGM. Η μελέτη της θεωρίας αυτής θα αποκαλύψει ενδιαφέρουσες ιδιότητές της, καθώς και τη σχέση της με άλλους φορμαλισμούς, όπως το μοντέλο θεμελιακής γνώσης (foundational model) και η θεωρία πλεγμάτων (lattice theory).

Στο τέταρτο μέρος της διατριβής, θα περιορίσουμε περαιτέρω την μελέτη μας ώστε να αναφέρεται στους κύριους φορμαλισμούς αναπαράστασης οντολογιών, δηλαδή στις Περιγραφικές Λογικές και στην γλώσσα OWL. Θα διατυπώσουμε εξειδικευμένες συνθήκες υπό τις οποίες τέτοιες λογικές είναι συμβατές (ή μη) με την θεωρία των AGM, καθώς και αποτελέσματα τα οποία επιτρέπουν τον καθορισμό της παραπάνω συμβατότητας (ή ασυμβατότητας) για λογικές που δεν μελετώνται σε αυτή την εργασία. Το κύριο αποτελέσμα αυτού του μέρους είναι ότι ορισμένες Περιγραφικές Λογικές, καθώς και η OWL, δεν είναι συμβατές με την θεωρία των AGM, ενώ σε άλλες η θεωρία μπορεί να εφαρμοστεί επιτυχώς για την ανάπτυξη αλγορίθμων εξέλιξης οντολογιών. Για τις τελευταίες, περιγράφεται επίσης ένας συμβατός με την θεωρία των AGM αλγόριθμος αφαίρεσης (contraction).

Επόπτης Καθηγητής:
Δημήτρης Πλεξουσάκης,
Αναπληρωτής Καθηγητής Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

To Roula,

for her love and encouragement

Acknowledgments

First of all, I would like to thank my advisor, Associate Professor Dimitris Plexousakis, for his support throughout the long years of this PhD. His unconditional trust on my abilities as well as his expert guidance and supervision always gave me the confidence necessary to continue when things were rough. Apart from his help on research issues, I am also grateful to him for the fact that I could always rely on his protection from any eventuality that would delay or distract me from my research.

A special thanks should also go to Professor Grigoris Antoniou, member of my supervising committee, for his cooperation with me and my supervisor in the development of a significant part of this work.

Furthermore, I would like to thank Professor Nicolas Spyratos, who was also a member of my supervising committee and was always there whenever his help was needed, as well as the members of my examination committee, Professor Panos Constantopoulos and Associate Professors Vassilis Christophides, Manolis Koubarakis and Pavlos Peppas for their helpful comments and criticism.

Moreover, I would like to acknowledge the support of the Institute of Computer Science (ICS-FORTH) both financially and in facilities, as well as the University of Crete for granting me the Maria Manasaki legacy's fellowship. Several people in ICS and the University helped me in different ways throughout this PhD, deserving special acknowledgement; a partial list includes Maria Moutsaki, Rena Kalaitzaki, Lida Harami and Yannis Tzitzikas.

I would also like to thank my friends (you know who you are) for all the happy times we had together and for always being there to remind me of the many joys of life.

This work would not have been possible without the unconditional and continuous support and encouragement from my mother, Maria, and my brother, Kostas, in both psychological and material terms. Their help was both invaluable and irreplaceable.

Last, but certainly not least, I would like to thank Roula Tzimpragou for always being there for me; her company acted as an instant reliever from all the frustrations, tiredness and disappointments that were part of this PhD. She was always there to inspire me and help me through the rainy days. This PhD is devoted to her.

Table of Contents

Chapter 1: Introduction **1**
1.1. General Setting 1
Ontologies and the Semantic Web 1
Ontology Change 1
1.2. Motivation and Main Ideas 3
Ontology Change Terminology 3
Belief Change and Ontology Evolution 3
The Generalization of the AGM Theory 4
The AGM Theory in the Ontological Context 6
1.3. Outline of this Dissertation 6

Chapter 2: Preliminaries **7**
2.1. Introduction 7
2.2. Ontologies and Ontology Change 7
Ontologies 7
Ontology Change 8
Ontology Mapping, Morphism, Alignment, Articulation and Translation 8
Ontology Evolution and Versioning 11
Ontology Integration and Merging 16
Summary of Ontology Change Subfields 19
2.3. Description Logics (DLs) and Web Ontology Language (OWL) 21
Ontology Representation Formalisms 21
Description Logics (DLs) 21
Semantics of DLs 23
Web Ontology Language (OWL) 24
Ontologies and DLs 24
2.4. Belief Change 25
Introduction 25
Basic Definitions 25
Representation Issues 25
Belief Change Operations 27
Semantics for the Change 28
Postulation Versus Explicit Construction 29
Issues Involved in the Change Process 29
Discussion 31
2.5. AGM Theory 31
Introduction 31
Tarski's Logical Framework 31
AGM Assumptions and Operations 33
The AGM Postulates 33
Criticisms and Importance 34
Results Based on the AGM Theory 35

Chapter 3: Belief Change in Ontologies 37
3.1. Introduction 37
3.2. Weaknesses of Existing Ontology Evolution Tools 37
The Current Situation 37
First Problem: Manual Ontology Evolution 39
Second Problem: Too Many Operators 40
Third Problem: Lack of Adequate Formalizations 41
3.3. Our Proposal 42
General Guidelines Towards a Solution 42
The Proposed Solution 42
Formalization: Basic Ideas 43
Representation of Ontologies 43
Change Representation Phase 46
Change Representation Phase: Facts Versus Modifications 50
Can I Remove the Penguin? 50
Closed Vocabulary Assumption and the Existence Assertion Operator 51
Open Vocabulary Assumption 54
Discussion on OVA and CVA 54
Defining an Ontology Evolution Algorithm 56
The Semantics of the Set of Changes 57
Foundations and Coherence Theories (Belief Bases and Belief Sets) 58
Postulation Methods Versus Explicit Constructions 59
Principle of Primacy of New Information 59
Principle of Irrelevance of Syntax 59
Consistency, Inconsistency, Coherence and Incoherence 60
Principle of Adequacy of Representation 64
Principle of Fairness 64
Principle of Minimal Change 65
Example of Application 66
3.4. Discussion and Comparison of the Approaches 72
3.5. Similar Approaches 73
Introduction 73
AGM Theory in Ontology Evolution 73
Epistemic Entrenchment in Ontology Evolution 74
Maxi-adjustment in Ontology Evolution 74

Chapter 4: Generalizing the AGM Theory 75
4.1. Introduction 75
4.2. Motivation and Goals 75
4.3. The Generalization of the AGM Theory 76
Basic Setting 76
Generalized Basic AGM Postulates for Contraction 77
4.4. AGM-Compliance and Decomposability 78
AGM-Compliance 78
Complement Beliefs and Decomposability 79
Decomposability and AGM-Compliance 82
4.5. AGM-Compliance and Lattice Theory 84
AGM-Equivalent Logics 84
AGM-Equivalence: Basic Properties 86
AGM-Equivalence and Decomposability 89
Lattice Theory 90
Using Lattices for Visualization 94
4.6. Alternative Characterizations: Cuts and Max-Cuts 97
Cuts 97
Max-cuts 99

4.7. Discussion on AGM-Compliance 102
AGM Assumptions and AGM-Compliance 102
Related Results and Discussion 105
Propagating AGM-Compliance 106
4.8. AGM Theory and the Foundational Model 109
Belief Base Operations and the AGM Theory 109
Base-AGM-Compliance 109
Base Decomposability 110
Base Cuts 114
Discussion on Base-AGM-Compliance 116
4.9. Related Results and Discussion 117
Roots 117
Supplementary Postulates 123
Generalized AGM Postulates for Revision 127
Topology of AGM-Compliance 133

Chapter 5: DLs, OWL and the AGM Theory 135
5.1. Introduction 135
5.2. DLs under the CVA Model 135
5.3. OVA Model: Conditions for AGM-Compliance 137
Introduction 137
Initial Thoughts and a Naïve Approach 137
The Need for Transformations 138
Defining the Transformations 139
Applications and Additional Results 141
Discussion 145
An AGM-compliant Contraction Operator 145
5.4. OVA Model: Conditions for Non-AGM-Compliance 146
Results 146
Discussion 149
5.5. OWL and AGM-Compliance 149
Introduction 149
OWL DL and OWL Lite Without Annotation Features 150
OWL With Annotation Features 151
OWL With “owl:imports(.)” 152
5.6. Discussion 153
Application to DLs in the Literature 153
DLs and Base-AGM-Compliance 155
Topology of AGM-Compliance with DLs and OWL 155

Chapter 6: Conclusion and Future Work 157
6.1. Belief Change and Ontology Change 157
Research Path 157
AGM-Compliance in Description Logics and OWL 158
6.2. Main Contributions of our Research 159
Contributions to the Field of Belief Change 159
Contributions to the Fields of Ontology Change and Ontology Evolution 160
Practical Contributions 160
6.3. Concluding Remarks 161
6.4. Future Work 163

References 165

List of Figures

Figure 2.1. Example of a modeling convention 18
Figure 3.1. The visualization of BIRD_ONTOLOGY 45
Figure 3.2. The PENGUIN_ONTOLOGY and the UNICORN_ONTOLOGY 63
Figure 3.3. Example of a revision (DUMBO_ONTOLOGY) 69
Figure 4.1. Intuition for decomposability 80
Figure 4.2. AGM-equivalence 86
Figure 4.3. A lattice and a poset 91
Figure 4.4. Lattices and logics 94
Figure 4.5. Visualizing Propositional Logic using a lattice 95
Figure 4.6. Decomposability in a lattice 96
Figure 4.7. Visual representation of example 4.1 96
Figure 4.8. A typical DL under CVA 97
Figure 4.9. Examples of cuts 98
Figure 4.10. Max-cuts do not always exist 100
Figure 4.11. Negation in Tarski's framework 103
Figure 4.12. A decomposable logic not satisfying the AGM assumptions 106
Figure 4.13. Lattices and logics (foundational case) 111
Figure 4.14. Examples of base cuts 114
Figure 4.15. Roots and fully independent root sets 118
Figure 4.16. Revision example 130
Figure 4.17. Maximality in revision 132
Figure 4.18. Topology of AGM-compliance 133
Figure 5.1. The intuition behind lemma 5.4 146
Figure 5.2. Topology of AGM-compliance with DLs and OWL 156

List of Tables

Table 2.1. Summary of the various subfields of ontology change 20
Table 3.1. Ontology evolution operations 47
Table 5.1. Transforming DL axioms into the form A⊒⊤ 140
Table 5.2. AGM-compliant and non-AGM-compliant DLs 154

Chapter 1: Introduction

Those who do not stop asking silly questions become scientists.

Leon Lederman

1.1. General Setting

Ontologies and the Semantic Web

It is often argued that current World Wide Web (WWW) will benefit from the introduction of semantic content in web pages [BLHL01]. So far, web pages have contained information related only to their appearance in a browser. Therefore, current WWW is "machine-readable" but not "machine-understandable". The addition of semantical information will allow a computer agent to interpret the content of the page as well. Such an ability will allow the development of more sophisticated search engines and other web services and will enhance a user's browsing experience and productivity. This development constitutes the vision of the Semantic Web and is currently one of the most active topics in Artificial Intelligence (AI) research.

Originally introduced by Aristotle, *ontologies* are formal models about how we perceive a domain of interest and provide a precise, logical account of the intended meaning of terms, data structures and other elements modeling the real world. As such, they are often viewed as the key means through which the Semantic Web vision can be realized and have already found several applications in the area of Knowledge Representation (KR) and in the Semantic Web. Ontologies are so important in the Semantic Web because they provide a means to formally define the basic terms and relations that comprise the vocabulary of a certain domain of interest [LE03], enabling machines to process information provided by human agents. As a result, ontologies can help in the representation of the content of a web page in a formal manner, so as to be suitable for use by an automated computer agent, crawler, search engine or other web service. The importance of ontologies in current AI research is also emphasized by the interest shown by both the research and the enterprise community to various problems related to ontologies and ontology manipulation [MFRW00].

Ontology Change

Ontologies are often large and complex structures, whose development and maintenance give rise to several sturdy and interesting research problems. One of the most important such problems is *ontology change*, which refers to the problem of changing an ontology in response to a certain need. Throughout this dissertation, the term ontology change will be

used in a broad sense, covering any type of change, including changes to the ontology in response to external events, changes dictated by the ontology engineer, changes forced by the need to translate the ontology in a different language or using different terminology and so on.

Several reasons for changing an ontology have been identified in the literature. An ontology, just like any structure holding information regarding a domain of interest, may need to change simply because the domain of interest has changed [SMSS03]; but even if we assume a static world (domain), which is a rather unrealistic assumption for most applications, we may need to change the perspective under which the domain is viewed [NK04], or we may discover a design flaw in the original conceptualization of the domain [PT05]; we may also wish to incorporate additional functionality, according to a change in users' needs [HS05a]. Furthermore, new information, which was previously unknown, classified or otherwise unavailable may become available or different features of the domain may become important [HHL99]. Moreover, ontology development is becoming more and more a collaborative and parallelized process, whose subproducts (parts of the ontology) need to be combined to produce the final ontology [KN03]; this process would require changes in each subontology to reach a consistent final state; but even then, the so-called "final" state is rarely final, as ontology development is usually an ongoing process [HHL99].

There are also reasons related to the distributed nature of the Semantic Web: ontologies are usually depending on other ontologies, over which the knowledge engineer may have no control; if the remote ontology is changed for any of the above reasons, the dependent ontology might also need to be modified to reflect possible changes in terminology or representation [HHL99]. In other cases, a certain agent, service or application may need to use an ontology whose terminology or representation is different from the one it can understand [EBBBB$^+$04], so he needs to perform some kind of translation (change) in the imported ontology. Last but not least, we may need to merge or integrate information from two or more ontologies in order to produce a more appropriate one for a certain application [PGPM99].

The problem of ontology change is far from trivial. Several philosophical problems related to the adaptation of knowledge in general have been identified in the research area of *belief change* (also known as *belief revision*) [Gar92a], [Gar92b], [KM90a]; most of them are also applicable to knowledge represented in ontologies. Moreover, the large size of modern day ontologies complicates this problem even further [MFRW00]. But it's not just that: the Semantic Web is characterized by decentralization, heterogeneity and lack of central control or authority. This is both a blessing and a curse; these features have greatly contributed to the success of the WWW (and constitute key features of the Semantic Web) but they have also introduced several new, challenging and interesting problems, which didn't exist in "traditional" AI.

As far as ontology change is concerned, one such problem is the lack of control on who uses a certain ontology once it has been published. Subtle changes in an ontology may have unforeseeable effects in dependent applications, services, data and ontologies [SMMS02]; ontology designers cannot know who uses which part of their ontology and for what purpose, so they cannot predict the effects of a given change on their ontology. The same holds in the opposite direction: if an ontology is depending on other ontologies, there is no way for the ontology designer to control when and how these ontologies will change. These facts raise the need to support and maintain different interoperable versions of the same ontology [HHL99], [KFKO02], [HS05b], a problem greatly interwoven with ontology change [KF01]. On the

other hand, heterogeneity leads to the absence of a standard terminology for any given domain which may cause problems when an agent, service or application uses information from two different ontologies [EBBBB+04]. As ontologies often cover overlapping domains from differing viewpoints and with differing terminology, some kind of translation may be necessary in many practical applications.

1.2. Motivation and Main Ideas

Ontology Change Terminology

In order to cope with the many different aspects of the problem of ontology change, several related research disciplines have emerged (such as ontology evolution, alignment, merging, mapping etc), each dealing with a different facet of the problem of ontology change. These areas are greatly interlinked; as a result, several works and systems deal with more than one of these topics causing a certain confusion to a newcomer. This confusion is further increased by the fact that certain terms are often used with differing meanings in the relevant literature, denoting similar, but not identical, research directions or concepts. For examples of such confusing and overused terms refer to [FP05], [PGPM99]; more examples will be presented in later chapters of this dissertation.

We believe that this lack of a standard terminology constitutes a major bottleneck for the ontology change community, causing an unnecessary confusion as well as misunderstandings. For this reason, one of the purposes of this work is the introduction of a terminology, following the most common uses of the various terms in the literature. Fixing this terminology will allow us to determine the boundaries of each field as well as to get a grip on their interactions and connections. Therefore, the first contribution of this thesis is a shallow, yet broad, literature review on the field of ontology change, as well as the introduction of a certain, broadly accepted terminology that will, hopefully, serve as a point of reference for the ontology change community. This part of our work has been published in [FP05].

Belief Change and Ontology Evolution

Our aforementioned study of the different types of ontology change will be done under a critical eye, uncovering certain weaknesses of current research. In particular, we will argue that the field of ontology change is characterized by the lack of adequate formalizations for the various processes involved [FPA06]. Most of the available methods for ontology change attempt to emulate human behavior [SM02], using certain heuristics which are heavily based on the expertise of their developers and ontology engineers. They are not theoretically founded and their formal properties remain generally unspecified. Furthermore, they require varying levels of user intervention in order to work, a rather unrealistic assumption for several applications for reasons that will be thoroughly analyzed in later sections (see also [FP05]).

These observations apply, at varying levels, to the field of ontology change as a whole; however, the focus of this thesis lies on *ontology evolution*, a certain subfield of ontology change which deals with the problem of modifying an ontology in response to a certain change in the domain of discourse or its conceptualization. Most of our argumentation, as well as the solution proposed, will be applicable to the problem of ontology evolution, even though it may, in some cases, be partly applicable to related fields as well.

The key idea of our attempt towards resolving the aforementioned deficiencies of current research on ontology evolution is to use the extensive work that has been performed in the

field of belief change and apply it to the ontology evolution context. Belief change deals with the adaptation of a Knowledge Base (KB) to new information [Gar92a]; viewing an ontology as a special type of KB allows us to view ontology evolution as a special case of belief change. Therefore, it makes sense to apply techniques, methods, tools, ideas, results and intuitions developed in the belief change literature to the field of ontology evolution.

To do that, we will study both fields, outlining their similarities and differences, explaining in detail why we consider them closely related and providing the necessary connections between them. In particular, we will propose the migration of methods developed by the belief change community to ontology evolution and underline the main problems that appear during this migration. Notice that this part of our work will be kept at a fairly abstract level and will not give any concrete solutions to the problem of ontology evolution; our aim at this point is to provide some general guidelines only towards a research direction that will ultimately give us the desired solutions. This analysis will also show that our approach can be used as a supplementary tool enhancing current ontology evolution systems; therefore, our research is actually supplementary, and not rivalrous, to currently performed research.

We will argue that our approach allows us to kill several birds with one stone, by solving several of the problems currently faced by the ontology evolution community. More specifically, the mature field of belief change will provide the necessary formalizations that can be used by the, yet immature [NK04], ontology evolution research. Belief change has always dealt with the automatic adaptation of a KB to new knowledge, without human participation in the process; the ideas and algorithms developed towards this aim may prove helpful in our effort to loosen up the dependence of the ontology evolution process on the knowledge engineer. Finally, previous work on belief change can protect us from potential pitfalls and prevent the "reinvention of the wheel" for problems whose counterparts have already been studied in the rich belief change literature, while belief change intuitions that are not directly applicable to ontology evolution may serve as an inspiration for developing solutions to similar problems faced by ontology evolution researchers.

The idea of using results from the belief change literature as an inspiration and influence for ontology evolution research has not been generally considered in the literature, notable exceptions being [KL04], [LM04], [MLB05] which present preliminary work on the subject. In section 3.5 we present these works and study their relationship with this thesis. Similar approaches can be found in [Foo95], [Wass98], [WF99] as well, which address the issue from a different standpoint, in which the evolving objects (and therefore the main objects of study) are the concepts. This viewpoint is quite different from ours, because, in our case, the evolving object is an ontology as a whole.

Therefore, the approach presented in this thesis is novel. The second contribution of this thesis is the introduction of this innovative research path; we hope that our abstract study on the transferability of belief change ideas to the ontology evolution context will provide the foundations upon which deeper results can be based, thus paving the road for the development of concrete solutions to the problem of ontology evolution. This part of our work has been published in [FP05], [FP06], [FPA05b], [FPA06].

The Generalization of the AGM Theory

It is obvious that exhaustively following the abstract research path we proposed in the last subsection is a very ambitious and long-term goal, because of the richness of the belief change literature; for the purposes of this thesis, we chose to narrow our research objectives

significantly, by restricting ourselves to a certain belief change theory, namely the AGM theory [AGM85]. This study will exploit the aforementioned abstract ideas and intuitions but will have a more specific character; in this respect, it could be viewed as an application of our more general objective.

The AGM theory was chosen because of its unparalleled influence in belief change research. This theory provided the theoretical foundations of the field by setting a stable and broadly accepted theoretical basis upon which most subsequent research on belief change was based. With this fact in mind, the AGM theory emerges as the most natural first choice for a belief change theory to be migrated in the ontology evolution context. More reasons for this choice will be presented later in this dissertation.

The purpose of this part of our work is twofold: firstly, it determines the feasibility of applying one of the most important belief change theories (the AGM theory) to the context of ontology evolution; secondly, it is exemplary of the more general research path we propose and shows the main problems that are likely to be encountered during the migration of belief change theories to the context of ontology evolution.

The application of the AGM theory to the ontological context will be made in two phases. Initially, we will observe that the AGM theory uses certain assumptions [AGM85] that restrict its application to classical logics only, like Propositional Logic (PL) and First-Order Logic (FOL) [End72], [Men87]; this, as we will see in section 4.2, overrules its direct application to several logics used for the representation of ontologies (see [FPA04a]), like Description Logics (DLs) [BCMNPS02] and OWL [DSBHH+04]. On the other hand, the intuitions behind the theory's development are independent of the underlying knowledge representation formalism.

This problem is typical of the problems encountered during the migration of belief change ideas to the ontology evolution context: the differences on the underlying intuitions are minimal, but the representation languages and formalisms used are quite different. In such cases, it makes sense to recast the theory under question (in this case, the AGM theory) in a setting general enough to contain ontology representation languages (like DLs and OWL).

Towards this direction, we will drop most AGM assumptions and reformulate the theory in a very general context that engulfs most knowledge representation languages (a notable exception being the family of nonmonotonic logics [Ant97]). This generalization can be of use in several contexts (including, but not limited to, ontology evolution), as it allows the use of the AGM theory in a great variety of formalisms.

Unfortunately, our study will show that this reformulation cannot be done in a perfect manner, because there are certain logics for which the reformulation, though possible theoretically, is not applicable in practice. We will characterize such logics using three different necessary and sufficient conditions which determine the applicability of the AGM theory in any given logic. Additionally, our work will result to a new representation theorem for the AGM theory, will uncover interesting connections of the AGM theory with lattice theory [Gra71] and provide a definite answer on the issue of the relation of the AGM theory with belief base algorithms [Han96] and the foundational model [Gar92b], among other things. These results constitute the third contribution of this thesis. This part of our work has been published in [Flou04], [FPA04a], [FPA04b], [FPA05b].

The AGM Theory in the Ontological Context

Given the theoretical foundations set by the results of the third part of this thesis, we will be able to determine the applicability of the AGM theory to languages used for representing ontologies. This determination constitutes the second phase of our reformulation of the AGM theory and the fourth contribution of this thesis. During this phase, we focus on two families of ontology representation languages which are expected to play a key role in the development of the Semantic Web, namely DLs and OWL [BHS03], [DSBHH+04].

Our research will uncover certain specialized conditions under which the AGM theory can (or cannot) be applied to a certain DL. We will present several positive and negative results related to this applicability, as well as certain empirical rules allowing one to determine this applicability for DLs not covered by this work. Similar (negative) results will be shown for OWL. These results determine, to a large degree, the applicability of the AGM theory in logics used for ontology representation and constitute the fourth contribution of this thesis. This part of our work has been published in [Flou05], [FPA05a], [FPA05c], [FPA05d].

1.3. Outline of this Dissertation

The rest of this dissertation is organized as follows:

- In the next chapter, we will perform a short literature review on the different types of ontology change (with emphasis on ontology evolution) in an attempt to draw a fine line between these closely related research fields and study their relation and interaction. This survey will include the fixing of a much needed terminology for this field. It will also provide the necessary background that will be of use in subsequent chapters and uncover the main weaknesses of the proposed approaches. Finally, for purposes of self-containment, we will briefly review the field of belief change, the AGM theory and Description Logics.
- In chapter 3, we will elaborate more on the problems of the current research and propose an alternative, more formal, logic-based method to address certain aspects of ontology evolution, which is inspired and greatly influenced by previous research on the area of belief change. We will argue that this logic-based approach complements the currently used approaches and could be used as a supplementary tool for current ontology evolution algorithms and systems.
- Chapter 4 constitutes the first step towards a more specific approach to the problem of ontology evolution, by describing the proposed generalization of the AGM theory to a large class of logics that includes the major formalisms used for ontological representation. This study will uncover several interesting results related to the AGM theory.
- In chapter 5, based on the theoretical foundations set in chapter 4, we will apply our results to DLs and OWL and show that the AGM theory is partly applicable to logics used for ontological representation.
- We will conclude in chapter 6 with some discussion, a summary of the results of this dissertation and the planned future work.

Chapter 2: Preliminaries

If I have seen farther, it is by standing on the shoulders of giants.

Sir Isaak Newton

2.1. Introduction

In this chapter we are going to present previous work on various subjects that are relevant to our research, namely ontology change, Description Logics, OWL, belief change and the AGM theory. Results from the literature that will be used throughout this dissertation will be presented and all the relevant terminology will be fixed; this will make this dissertation self-contained. As explained before, given the vagueness of the terminology used in the ontology change field, the introduction of a terminology on ontology change (section 2.2) serves the additional purpose of providing a certain proposal that will hopefully clear the picture regarding the exact meaning of each term in the literature. Our survey of the various research areas in this chapter will be broad, but will not get into the details of each field unless necessary. Its purpose is to give a clear overall picture of each relevant area; the interested reader is referred to the various bibliographic references that will appear throughout this chapter for details or deeper results.

2.2. Ontologies and Ontology Change

Ontologies

Obviously, one of the most important concepts that will be used in this dissertation is the concept of an ontology. The term ontology has come to refer to a wide range of formal representations, including taxonomies, hierarchical terminology vocabularies or detailed logical theories describing a domain [NK04]. For this reason, a precise definition of the term is rather difficult. One definition that is commonly used in the literature was given in [Gru93] where an ontology was defined to be a *specification of a shared conceptualization of a domain*.

A more formal, algebraic, approach, identifies an ontology as a pair <S,A>, where S is the *vocabulary* (or *signature*) of the ontology (being modeled by some mathematical structure, such as a poset, a lattice or an unstructured set) and A is *the set of ontological axioms*, which specify the intended interpretation of the vocabulary in a given domain of discourse [KS03].

Ontology Change

As already mentioned, we will use the term ontology change in a broad sense to cover all aspects of ontology modification, as well as the problems that are indirectly related to the change operation such as the maintenance of different versions of an ontology or the translation of ontological information in a common terminology. In short, we will use the term ontology change to refer to the problem of *deciding the modifications to perform upon an ontology in response to a certain need for change as well as the implementation of these modifications and the management of their effects in depending data, services, applications, agents or other elements*.

In this definition, the decision on the modifications to perform may be made automatically, semi-automatically or manually; the implementation of the chosen modifications may (but need not) involve keeping a copy of the original ontology. The "need" to change the ontology may take several different forms, including, but not limited to, the discovery of new information (which could be some instance data, another ontology, a new observation or other), a change in the focus or the viewpoint of the conceptualization, information received by some external source, a change in the domain (i.e., a dynamic change in the modeled world), communication needs between heterogeneous sources of information, the fusion of information from different ontologies and so on.

This definition covers several related research areas which are studied separately in the literature. In this thesis, we identify nine such areas, namely *ontology mapping, morphism, alignment, articulation, translation, evolution, versioning, integration* and *merging*. Each of these areas deals with a certain facet of the complex problem of ontology change from a different view or perspective, covering different application needs, change scenarios or "needs for change". The diversity of the different forms that the problem of ontology change may take disallows any attempt for a unifying formal specification, because the purpose, the scope, the input, the output and the properties of the various fields differ. For more details on this issue refer to table 2.1 at the end of this section.

Despite the differences however, these areas are greatly interlinked and several papers, works and systems deal with more than one of these problems. In other cases, the same term is used in different papers to describe different research areas. This situation can easily lead to misunderstandings, confusion and unnecessary waste of effort, especially for a newcomer. In the following sections, we will attempt to precisely define the boundaries of each area and uncover their relations and differences. This attempt will hopefully draw a fine line between the various research areas, allowing the clarification of the meaning of each term and making the differences and similarities between them explicit. The definitions provided here will be based on the most common uses of each term in the literature.

Ontology Mapping, Morphism, Alignment, Articulation and Translation

Works related to these areas try to mitigate the problems caused by the heterogeneity of the Semantic Web. The general motivation for these works is that different sources of information generally use different terminology, different representation languages and different syntax to refer to the same or similar concepts. A nice list of use cases where this heterogeneity may cause problems can be found in [EBBBB+04].

The solution that is generally proposed for the above problem consists of a set of "translation rules" of some kind that allow us to nullify the differences in terminology or syntax. To put it simply, the goal of the whole process is to make two ontologies refer to the

same entities with the same name and to different entities with different names. For example, an ontology alignment algorithm should be able to identify that two concepts (classes) RESEARCHER and RESEARCH_STAFF_MEMBER that appear in two different ontologies refer to the same real-world concept, i.e., the class of all researchers. It should also be able to differentiate between two different uses of the entity CHAIR, as it could refer to the class of chairs (as a furniture) in one ontology and to the people forming a Workshop's Chair in another ontology.

Therefore, these areas basically deal with the same problem (i.e., how to deal with heterogeneous information); however, the output of the methods (i.e., the result that is produced towards the solution of the problem) is different in each case. The different research fields are identified based on the type of "translation rules" that is produced as the output. Due to the close relationship between these topics, sometimes the term ontology alignment (in [EBBBB$^+$04]) or ontology mapping (in [KS03]) is used to refer collectively to all these areas. In the following, we will provide a definition of these fields and outline their differences and similarities. Most of the material and the definitions for this subsection are taken from [EBBBB$^+$04] and [KS03].

The term *ontology mapping refers to the task of relating the vocabulary of two ontologies that share the same domain of discourse in such a way that the mathematical structure of ontological signatures and their intended interpretations, as specified by the ontological axioms, are respected*. The result of an ontology mapping algorithm is a collection of functions (morphisms) of ontological signatures. We could differentiate between two different types of mappings, namely the *total ontology mappings* and the *partial ontology mappings*. A total ontology mapping maps the whole source ontology (say $O_1=<S_1,A_1>$) to the target ontology O_2 while a partial ontology mapping maps a subontology of O_1 (say $O_1'=<S_1',A_1'>$, with $S_1' \subseteq S_1$ and $A_1' \subseteq A_1$) to O_2.

This definition restricts the mappings to ontological signatures. A more ambitious and interesting approach would be to create mappings that deal with both the signatures and the axioms of the ontologies. The term *ontology morphism* refers to that approach, i.e., *the development of a collection of functions (morphisms) that relate both ontological signatures and axioms*. Notice that ontology morphism, unlike the other fields discussed in this subsection, is not restricted to the ontology vocabulary only, but covers the ontological axioms as well.

In both ontology mapping and ontology morphism, we try to relate the two ontologies via functions. Alternatively, the two ontologies could be related in a more general fashion, namely by means of a relation. *The task of finding relationships between (vocabulary) entities belonging to two different ontologies is called ontology alignment*. So the result of ontology alignment is a binary relationship between the vocabularies of the two ontologies. This approach is more liberal, allowing greater flexibility, so it is more commonly used in practice.

A binary relationship could be decomposed into a pair of total functions from a common intermediate source; therefore, the alignment of two ontologies (say O_1, O_2) could be described by means of a pair of ontology mappings from a common intermediate ontology (say O_0). We use the term *ontology articulation* to refer to *the process of determining the intermediate ontology (O_0) and the two mappings to the initial ontologies (O_1, O_2).*

Finally, the term *ontology translation* is used in the literature to describe two different things. Under one understanding, ontology translation refers to *the process of changing the*

formal representation of the ontology from one language to another (say from OWL [DSBHH+04] to RDF [MSB06]). This changes the syntactic form of the axioms, but not the vocabulary of the ontology; thus, works related to ontology translation (under this understanding) leave the vocabulary of the ontology unaffected, dealing with the ontological axioms only. Under the second understanding, ontology translation refers to *a translation of the vocabulary*, in a manner similar to that of ontology mapping. The difference between ontology mapping and ontology translation is that the former specifies the functions that relate the two ontologies' vocabularies, while the latter applies these functions to actually implement the mapping.

As was made clear in the definitions of this subsection, the aforementioned research areas try (with a few exceptions) to relate the vocabularies of two ontologies. This problem is far from trivial; methods that are commonly used for this purpose include studying the taxonomic or mereological structure of the entities, evaluating name similarities (in effect, studying the string similarity of the entities' names, under a certain metric) and so on. Other methods use a thesaurus to study the linguistic similarities of terms (names), use semantic approaches, or determine the similarity based on the instances of each entity. The final similarity evaluation may also be affected by the evaluation of the similarity of the entities' neighborhood. In real systems, a combination of some of these approaches with some kind of human intervention usually works best. A detailed classification and description of these methods can be found in [EBBBB+04].

Two of the most important systems that deal with heterogeneity are PROMPT [NM00], [NM99a] (originally called SMART [NM99b]) and Chimaera [MFRW00]. In [Fer04], the term ontology matching is used to refer to an ontology mapping algorithm based on the linguistic properties of terms, using a thesaurus based on WordNet [Mil95]. In [SSK05], a certain string metric is proposed to evaluate name similarities of elements in different ontologies, upon which an ontology alignment algorithm could be based. Some thoughts on the issue of heterogeneity in the context of the SHOE language can be found in [HHL99], [HH00]. An interesting method of improving the results of an alignment process, which exploits user validation combined with machine learning techniques, can be found in [ESS05]. In [MNJ05], a probabilistic technique is used towards this aim; in that work, the final similarity evaluation of an ontology mapping algorithm is affected by the similarity probabilities of each entity's neighborhood, improving the initial mapping result. Another method based on probabilistic analysis, which takes into account uncertainty issues in the mapped ontologies can be found in [PDYP05]. A general-purpose approach to the problem of translation is described in [Cha00]. A much more extensive list of systems and works related to these research areas can be found in [BMRME04], [EBBBB+04], [KS03]; an evaluation of such works appears in [AGY05].

Unfortunately, heterogeneity resolution in ontologies still relies on human intervention; however, the process has to be automatic in order to be practical [KS03]. In this direction, advances in the field of natural language processing will probably help researchers gain a better understanding on the processes behind automatic heterogeneity resolution [KS03].

But why do we study these fields in a survey related to ontology change? The result of an ontology mapping algorithm (for example) is a mapping between the vocabularies of two ontologies instead of a modified ontology. Why does ontology mapping constitute a type of ontology change? The answer to these questions is twofold: firstly, heterogeneity resolution constitutes a prerequisite for successful ontology change, as it makes no sense to try to change

an ontology in response to new information unless both the ontology and the new information are formulated using the same terminology, language and syntax. So, it makes practical sense to study these fields along with the problem of ontology change. But it's not just that. We argue that ontology mapping (and the related fields) are not simply "closely related" to ontology change for "practical" reasons; on the contrary, they fall under the definition of ontology change, in the wide sense of the term that we use in this work.

Indeed, consider two agents with heterogeneous ontologies that need to communicate and an ontology mapping algorithm that provides a mapping allowing this communication. In this particular example, the driving force (need) behind ontology mapping is the need for communication. The mapping produced does not directly modify any ontology; however, it allows each agent to "change" the other agent's ontology to fit his own terminology, language and syntax. So the "change" in this particular case is made on-the-fly by each agent during each message exchange and it is trivial, given the mapping. In this sense, we could consider an ontology mapping algorithm to be an ontology change algorithm that provides us with a method to change an ontology (but does not perform the change itself); this change is made in response to a certain need, which in this particular case is the need for communication between agents with heterogeneous ontologies. Thus, ontology mapping is a part of ontology change in a very real sense. Similar arguments hold for the other fields discussed in this subsection.

Ontology Evolution and Versioning

In some works (e.g., [HS04]), ontology versioning is considered a stronger variant of ontology evolution. Under that viewpoint, ontology evolution is concerned with the ability to change the ontology without losing data or negating the validity of the ontology, while ontology versioning should additionally allow access to different variants of the ontology. Thus, while ontology evolution is concerned with the validity of the newest version, ontology versioning additionally deals with the validity, interoperability and management of all previous versions, including the current (newest) one.

This viewpoint is influenced by related research on relational and object-oriented database schema evolution and versioning [BKKK87], [KC88], [PO97]. A recent survey on this issue, studying the differences and similarities of the two fields can be found in [NK04], where ontologies are compared with database schemas outlining their differences and these differences' impact with respect to (ontology and database schema) evolution and versioning. In the same paper it is argued that, under this understanding, ontology evolution and versioning become indistinguishable; due to the distributed and decentralized nature of the Semantic Web, multiple versions of ontologies are bound to exist and must be supported. Furthermore, ontologies and dependent elements are likely to be owned by different parties or spread across different servers; as a result, some parties may be unprepared to change and others may even be opposed to it [HHL99]. All these facts force us to maintain and support different versions of ontologies, making ontology evolution (under this understanding) useless in practice.

Under a similar standpoint, the evolution of an ontology represents the evolution of our understanding of the domain of interest. Thus, a newer version of an ontology does not invalidate the old one, but "replaces" it in terms of usability (only); as a result, the different versions of an evolving ontology should coexist. This is reminiscent of the viewpoint employed in temporal databases [TCGJSS93]. The adaptation of this viewpoint makes

ontology versioning the only valid method of changing an ontology, while ontology evolution is an internal part of ontology versioning, dealing only with the determination of the next version.

We believe that the issue of modifying the ontology (ontology evolution) should be clearly separated from the issue of maintaining the interoperability of the different versions of the ontology that occur because of these changes (ontology versioning). This distinction is not always clear in the literature, because the extensive web of interrelationships that is usually formed around an ontology forces us to consider the issue of propagating the changes to dependent elements [MMSRV03]. This tight coupling has caused ontology evolution algorithms (and systems) to deal with these issues as well. For example, in [SMMS02], ontology evolution is defined as the timely adaptation of an ontology to changed business requirements, to trends in ontology instances and patterns of usage of the ontology-based application, as well as the consistent management and propagation of these changes to dependent elements. This definition covers both ontology evolution and versioning. Under this understanding, ontology versioning and change propagation are considered internal parts of the process of ontology evolution.

These viewpoints are partly justified by the fact that ontology evolution and ontology versioning are closely related areas, for reasons explained before; despite that, our definitions will strictly separate the two areas. In this thesis, we will use the term *ontology evolution to refer to the process of modifying an ontology in response to a certain change in the domain or its conceptualization* (for a more formal definition of an ontology evolution algorithm, refer to section 3.3). On the other hand, *ontology versioning refers to the ability to handle an evolving ontology by creating and managing different variants (versions) of it* [KF01]. In other words, ontology evolution is restricted to the process of modifying an ontology while maintaining its validity, while ontology versioning deals with the process of managing different versions of an evolving ontology, maintaining interoperability between versions and providing transparent access to each version as required by the accessing element (data, service, application or other ontology).

Ontology evolution could be considered as the "purest" type of ontology change, in the sense that it deals with the changes themselves. As already mentioned, this area will be the main focus of this thesis. Ontology evolution is a very important problem, as the effectiveness of an ontology based application heavily depends on the quality of the conceptualization of the domain by the underlying ontology [SMSS03], which is directly affected by the ability of an evolution algorithm to properly adapt the ontology to changes in the domain and to new needs in the conceptualization. Its importance is further emphasized by the fact that ontologies often model dynamic environments [SMMS02].

As already stated, an ontology is, according to [Gru93], a specification of a shared conceptualization of a domain. Thus, a change may be caused by either a change in the domain, a change in the conceptualization or a change in the specification [KF01]. The third type of change (change in the specification) refers to a change in the way the conceptualization is formally recorded, i.e., a change in the representation language. This type of change is dealt with in the field of ontology translation, studied in the previous subsection. Thus, our definition of ontology evolution covers the first two types of change only (changes in the domain and changes in the conceptualization).

Both types of changes are not rare. The conceptualization of the domain may change for several reasons, including a new observation or measurement, a change in the viewpoint or usage of the ontology, newly-gained access to information that was previously unknown, classified or otherwise unavailable and so on. The domain itself may also change, as the real world itself is generally not static but evolves over time. More examples of reasons initiating changes can be found in [KF01], [NK04].

In order to tame the complexity of the problem, six phases of ontology evolution have been identified, occurring in a cyclic loop [SMMS02]. Initially, we have the *change capturing* phase, where the changes to be performed are identified. Three types of change capturing have been identified: structure-driven, usage-driven and data-driven [HS04]. Once the changes have been determined, they have to be properly represented in a suitable format during the *change representation* phase. The third phase is the *semantics of change* phase, in which the effects of the change(s) to the ontology itself are identified; during this phase, possible problems that might be caused in the ontology by these changes are also determined and resolved. The *change implementation* phase follows, where the changes are physically applied to the ontology, the ontology engineer is informed on the changes and the performed changes are logged. These changes need to be propagated to dependent elements; this is the role of the *change propagation* phase. Finally, the *change validation* phase allows the ontology engineer to review the changes and possibly undo them, if desired. This phase may uncover further problems with the ontology, thus initiating new changes that need to be performed to improve the conceptualization; in this case, we need to start over by applying the change capturing phase of a new evolution process, closing the cyclic loop. An alternative, but similar, approach which identifies five phases, can be found in [PT05].

We will try to illustrate the role of the six phases using an example. Suppose that, for some reason, we decide to remove concept A from our ontology. During the change capturing phase we identify the need to remove A and initiate the six-phase process of ontology evolution.

During the change representation change we determine the kind of change(s) that must be performed in order to remove A. There are two major types of changes, namely elementary and composite changes [SMMS02] (also called atomic and complex in [SK03]). Elementary changes represent simple, fine-grained changes; composite changes represent more coarse-grained changes and can be replaced by a series of elementary changes. Even though possible, it is not generally appropriate to use a series of elementary changes to replace a composite change, as this might cause undesirable side-effects [SMMS02]. The proper level of granularity should be identified at each case. Examples of elementary changes are the addition and deletion of elements (concepts, properties etc) from the ontology. In our particular example, a simple “Delete_Concept” operation (which is an elementary operation) should be enough to perform the required change. This is not always the case though. For example, we might have wished to move the concept A to some other point in the concept hierarchy. This is represented by a composite change (or by a series of elementary changes). There is no general consensus in the literature on the type and number of composite changes that are necessary. In [SMMS02], 12 different composite changes are identified; in [NK04], 22 such operations are listed; in [SK03] however, the authors mention that they have identified 120 different interesting composite operations and that the list is still growing! In fact, the number of definable composite operations can only be limited by setting a granularity threshold on the operations considered; if we allow unlimited granularity, we will be able to define more and more operations of coarser and coarser granularity, limited only by our imagination [KN03].

Thus, creating a complete list of composite operations is not possible, but, fortunately, it is not necessary either, since a composite operation can always be defined as a series of elementary operations [KN03].

Once the required change is identified (i.e., "Delete_Concept" for the concept A in our example), we can proceed to the next step of the evolution process, which is the semantics of change phase. At this point, we should identify any problems that will be caused when the chosen action is actually implemented, thus guaranteeing the validity of the ontology at the end of the process. In our example, we need (among other things) to re-classify the instances of the concept A by determining what to do with those instances; for example, we could delete them or re-classify them to one of A's superconcepts. In [SMMS02], the authors suggest that the final decision should be made indirectly by the ontology engineer, through the selection of certain pre-determined "evolution strategies", which indicate the appropriate action in such cases. Other (manual or semi-automatic) approaches are also possible (see [HS04]). This phase is probably the most crucial of ontology evolution, because during that phase the (direct or indirect) changes caused by a given change request are determined.

During the implementation phase, the changes identified in the two previous phases are actually implemented in the ontology, using an appropriate tool, like, for example, the KAON API [SMMS02]. Such a tool should have transactional properties, based on the ACID model, i.e., guaranteeing Atomicity, Consistency, Isolation and Durability of changes [HS04]. It should also present the changes to the ontology engineer for final verification and keep a log of the implemented changes [HS04].

The change propagation phase should ensure that all induced changes will be propagated to the interested parties. In [MMSRV03], two different methods to address the problem are compared, namely push-based and pull-based approaches. Under a push-based approach, the changes are propagated to the dependent ontologies as they happen; in a pull-based approach, the propagation is initiated only after the explicit request of each of the dependent ontologies. In both [MMSRV03] and [SMMS02] the authors choose to use the former approach (push-based). Alternatively, one could avoid this step altogether, by using an ontology versioning algorithm [KFKO02], allowing the interested parties to work with the original version of the ontology and update to the newer version at their own pace, if at all. This alternative is considered more realistic for practical purposes, given the decentralized and distributed nature of the Semantic Web [HHL99].

Finally, the change validation phase should allow the ontology engineer to review and possibly undo the changes performed, or initiate a new sequence of changes to further improve the conceptualization of the domain as represented by the ontology.

The chosen model for ontology evolution allows us to ignore heterogeneity issues. Obviously, any ontology evolution method will collapse in the face of heterogeneous information, unless coupled with an algorithm like those discussed in the previous subsection. However, heterogeneity is not an issue under the proposed model, because ontology evolution algorithms assume human participation in the process; the ontology engineer identifies the changes to be performed during the change representation phase, so it can be reasonably assumed that these changes will be represented in a suitable terminology.

A list of systems related to the task of ontology evolution can be found in [HS04]. Some of these systems are simple ontology editors, but the most sophisticated of them provide features

that greatly aid the user in the task of changing an ontology. For more details on such systems refer to [DSWKB00], [HS04], or section 3.2 in this thesis.

Once the actual changes have been performed, ontology versioning algorithms come into play. Ontology versioning typically involves the storage of both the old and the new version of the ontology and takes into account identification issues (i.e., how to identify the different versions of the ontology), the relation between different versions (i.e., a tree of versions resulting from the various ontology modifications) as well as some compatibility information (i.e., information regarding the compatibility of any pair of versions of the ontology).

Several non-trivial problems are associated with this task. For example, any ontology versioning algorithm should be based on some type of identification mechanism to differentiate between various versions of an ontology. This task is not as easy as it may seem; for example, it is not clear when two ontologies constitute different "versions". Should any change in the file that stores the ontology specification constitute the creation of a new version? When a concept specification changes, but the new specification is semantically equivalent to the original one, should this constitute a new version? More generally, when the ontology changes syntactically, but not semantically, should this constitute a new version? These and similar problems are dealt with in [HHL99] and [KFKO02].

Another desirable property of an ontology versioning system is the ability to allow transparent access to different versions of the ontology, by automatically relating versions with data sources, applications and other dependent elements [KF01]. Other issues involved is the so-called "packaging of changes" [KFKO02] as well as the different types of compatibility and how these are identified [KF01].

Another related problem is the introduction of a certain "version relation" between ontological elements (such as classes) appearing in different versions of the ontology and the properties that such a relation should have. This relation is called a change specification in [KF01] and its role is to make the relationship between different versions of ontological elements explicit. Using this relation, one can identify the changes that any given element went through between different versions; in addition, a version relation should include certain meta-data regarding these changes [KFKO02]. In [PT05] this relation is stored using a "version log" which is actually a "version ontology" storing the different versions of each element, as well as the relation between them and some related meta-data, in a specially designed ontology. Similar considerations led to the definition of "migration specifications" [ZZLZY03], which associate concepts between different versions of an ontology after a change has been performed, as well as to the introduction of a temporal logic approach that allows the determination of the changes between ontologies through logical queries [HS05b].

As an aid to the task of ontology versioning, certain tools have been developed which automatically identify the differences between ontology versions; unfortunately, most such tools provide information at the level of basic (elementary) changes [KN03]. PROMPTDIFF [NKKM04] uses certain heuristics to compare different versions of ontologies and outline their differences, by producing a "structural diff" between them; OntoView [KFKO02] contains a tool similar to PROMPTDIFF, whose output is a certain "ontology of changes", i.e., an ontology that stores the types of changes performed, thus providing a kind of "mapping" between versions. [KN03] contains an excellent survey on the different ways that can be used to represent a set of changes, as well as the relation and possible interactions between such representations; furthermore, a standard "ontology of changes" is proposed,

containing both basic (i.e., elementary) and complex (i.e., composite) operations. A similar "ontology of changes" is proposed in [PT05], where the changes are identified through a version log stored in this "ontology of changes". A method to identify compatibility between ontology versions is presented in [HHL99], [HH00] where the SHOE language [LSRH97] is used to make backward compatibility between versions explicit in a machine-readable format, allowing a computer agent to determine compatibility between versions. This is an indirect approach to the problem of ontology versioning, as it allows the computer agent to determine autonomously which version to use, as opposed to [KF01], [KFKO02], where a more direct and centralized path is taken. In [HS05b], a temporal logic approach is used to allow access and reasoning in different versions of an ontology.

Ontology Integration and Merging

In short, both ontology integration and ontology merging refer to the development of a new ontology based on the information found in two or more source ontologies. As we will see however, the two terms refer to slightly different research areas. Despite that, the exact meaning of each term is not clear in the literature, as they are often used interchangeably. This situation has led to a certain confusion regarding the exact boundaries of each field.

In [NM00], [NM99a] ontology merging is defined as the process of creating a new, coherent ontology that includes information from two or more source ontologies. In these papers, ontology merging is implicitly assumed to include the process of resolving any possible heterogeneities between the merged ontologies, even though this is not apparent in their definition. Under that understanding, the difference between ontology merging and alignment is that ontology merging results in the creation of a new ontology, while in ontology alignment the merged ontologies persist, with links established between them. A similar use of the term can be found in [MFRW00], while, in [HHL99], the same research area is described using the term ontology integration. According to [LM04], ontology merging amounts to making sure that different agents use the same terms in identical ways (in a manner similar to ontology alignment). In [KS03] ontology integration is defined as the process of combining ontologies to build new ones, but whose respective vocabularies are usually not interpreted in the same domain of discourse.

Here we will define these terms along the lines of [PGPM99], which was an attempt to disambiguate between different uses of the term ontology integration. Three different uses of the term were identified in that paper. The first refers to the *composition (via reuse) of ontologies covering loosely related domains (subjects)*; this is mainly used when building a new ontology that covers all these subjects. The term *ontology integration* has been reserved for this process. The second use of the word refers to the *combination of ontologies covering highly overlapping or identical fields*; this process is used to fuse ontologies that contain information about the same subject into one large (and hopefully more accurate) ontology. The term *ontology merging* was attached to this interpretation. Finally, the third use of the term integration refers to the development of an application that uses one or more ontologies; the more appropriate term *ontology use* was reserved for this process. In this thesis, we will be interested in the first two understandings of the term only, namely ontology integration and merging.

There are certain subtle differences between the processes of integration and merging. Ontology integration is mainly applied when the main concern is the reuse of other ontologies. The domain of discourse of the new ontology is usually more general than the

domain of any of the source ontologies and integration often places the different (source) ontologies in different modules that comprise the resulting ontology. On the other hand, in ontology merging, the focus is on creating an ontology that combines information on a given topic from different sources. In this case, the information from the source ontologies is greatly intermingled, so it's difficult to identify the part(s) of the final ontology that resulted from each source ontology. A more extensive discussion on the subject can be found in [PGPM99].

It is a common practice in the literature (see for example [BMRME04], [HHL99], [NM00], [PGPM99]), to consider heterogeneity resolution to be an internal part of ontology merging or integration. This is a reasonable choice, because in most cases the fused ontologies come from different sources, so they are generally heterogeneous in terms of vocabulary, syntax, representation etc. Therefore, the task of relating the heterogeneous elements of the source ontologies constitutes a major part of the task of ontology merging (or integration). This is mostly true in merging, where the domain of discourse is (almost) identical. This has led to even more confusion on the exact meaning of the terms, as several works consider ontology merging (or integration) and alignment to be variations of the same problem (see for example [LM04], [NM00]).

However, it should be clear that simply resolving the heterogeneity issues between two ontologies is not sufficient for successful integration (or merging); recall that different ontologies may encode different viewpoints regarding the real world, thus several conceptual differences are bound to exist, even if the same terminology is used. This is reminiscent of how beliefs held by different people are often different (and in some cases contradictory), even if a common terminology is agreed upon. Similarly, modeling conventions and choices may be different; one example of modeling choice that often depends on personal taste or convention is whether to model a certain distinction between similar elements by introducing separate classes or by introducing a qualifying attribute relation in one class [Cha00] (see figure 2.1). Such modeling differences need to be taken into account when selecting what to keep from each ontology during the integration or merging process. Reckless inclusion of ontology elements from the source ontologies (even when homogeneous) is likely to lead to a problematic, invalid or inconsistent ontology. For this reason we argue that ontology merging (and integration) are not simply a variation of ontology alignment.

According to [Cha00], the process of merging can be broken down in five different steps. During the first step, we identify the semantic overlap between the source ontologies; during the second, we devise ways (transformations) that will bring the sources into mutual agreement in terms of terminology, representation etc. In the third step, we apply these transformations, so we can now take the union of the sources, which is the fourth step of the process. The final step consists of evaluating the resulting ontology for consistency, uniformity, redundancy, quality of conceptualization etc; this evaluation might force us to repeat some or all of the above steps. We believe that the same generic steps are necessary for integration as well, but they should be adapted to the different properties of the process. The author of [Cha00] claims that his translation tool is a necessary part of any merging algorithm, as it facilitates the design and implementation of the transformations used in the merging process (second and third step, respectively).

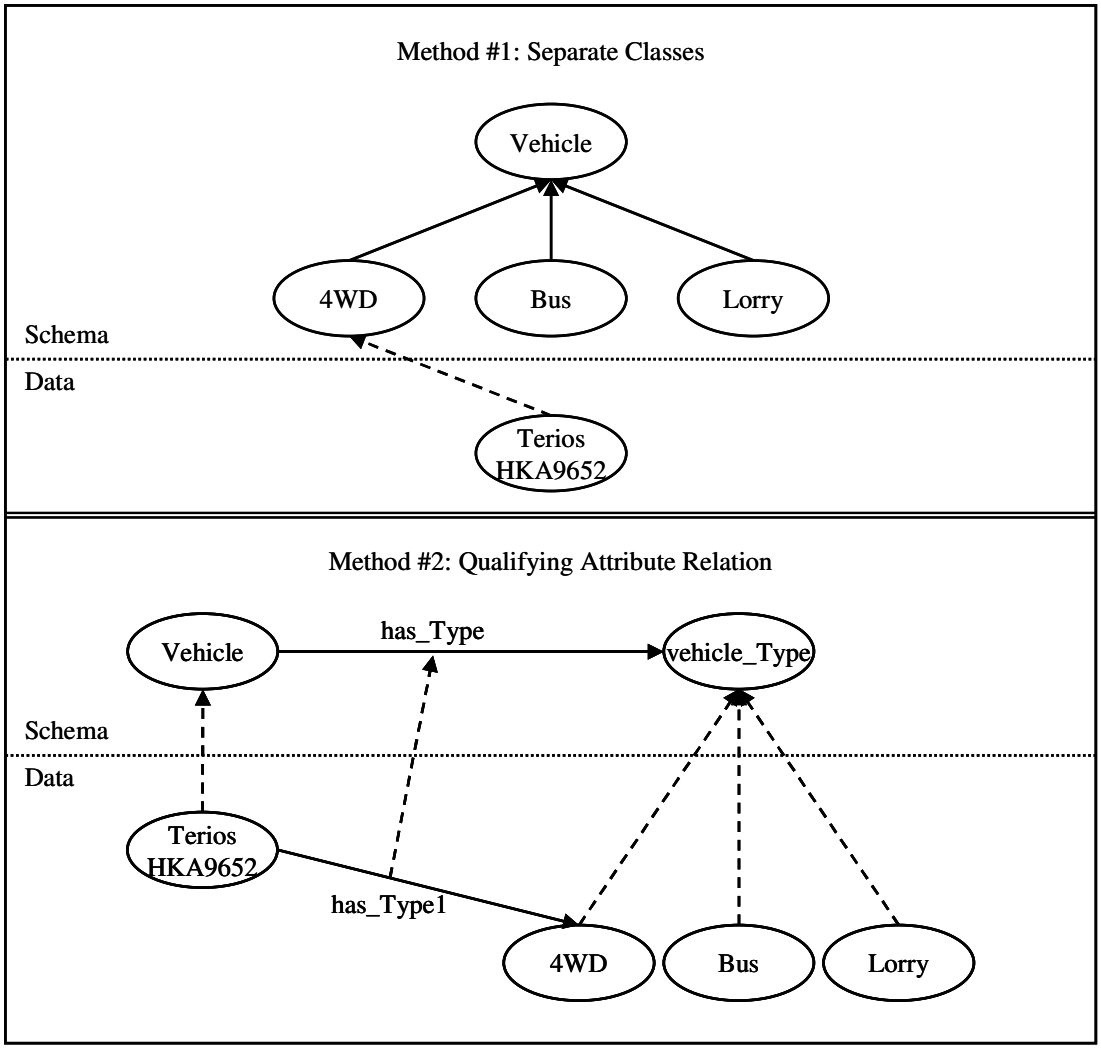

Figure 2.1. Example of a modeling convention

An extensive list of works related to ontology merging can be found in [BMRME04]. The main tools used for ontology merging are PROMPT [NM00] and Chimaera [MFRW00]. These tools use a semi-automatic approach focused on suggesting how elements from the source ontologies should be merged in the resulting ontology. The final choice relies on the ontology engineer. Some ideas on ontology merging (called integration there) in the context of the SHOE language can be found in [HHL99]; however, this work is focused on the part of merging that deals with heterogeneity resolution. An interesting theoretical approach to ontology merging can be found in [BCM99], while in [NM99a] some interesting connections of object-oriented programming with the problem of ontology merging are uncovered. The FCA-MERGE algorithm [SM02] performs ontology integration in a very efficient way, but is based on certain strong assumptions. A more detailed list of tools and systems related to the problem can be found in [PGPM99]. It should be emphasized that all the currently available tools use manual or semi-automatic methods to perform merging (or integration), guiding the user through the different steps of the process via some intuitive interface. The automatic processes behind ontology merging and integration are not yet well-understood: in [PGPM99], it is claimed that "ontology merging is currently more of an art".

Even though the issue of evaluating ontology merging techniques is still open in AI [SM02], certain comparison attempts have been made. In [LE03], the authors perform a

comparison between PROMPT and Chimaera in the context of bioinformatics. In [NM00], the same two tools are compared with the generic Protégé-2000. Furthermore, [MFRW00] compares the efficiency of ontology merging with a simple plain-text editor, merging with the Ontolingua editor and merging with the specialized tool Chimaera, which is described in the same paper. These comparisons are made from a certain standpoint; a general, objective comparison is difficult, as it is not clear how the utility of such tools could be measured [MFRW00].

Summary of Ontology Change Subfields

In the previous subsections, we presented nine different research areas that comprise ontology change. In order to fully clarify the meaning of the different terms and the boundaries of each area, we present here a table (table 2.1, see next page), that summarizes the most important characteristics of each field. In this table, we describe the need for change that motivates each field (purpose), the expected input of an algorithm that deals with the problem (input), its expected output (output), as well as certain comments on the desired properties of such an algorithm (properties).

Ontology Mapping	**Purpose:**	Heterogeneity resolution, interoperability of ontologies
	Input:	Two (heterogeneous) ontologies
	Output:	A mapping between the ontologies' vocabularies
	Properties:	The output identifies related vocabulary entities
Ontology Morphism	**Purpose:**	Heterogeneity resolution, interoperability of ontologies
	Input:	Two (heterogeneous) ontologies
	Output:	Mappings between the ontologies' vocabularies and axioms
	Properties:	The output identifies related vocabulary entities and axioms
Ontology Alignment	**Purpose:**	Heterogeneity resolution, interoperability of ontologies
	Input:	Two (heterogeneous) ontologies
	Output:	A relation between the ontologies' vocabularies
	Properties:	The output identifies related vocabulary entities
Ontology Articulation	**Purpose:**	Heterogeneity resolution, interoperability of ontologies
	Input:	Two (heterogeneous) ontologies
	Output:	An intermediate ontology and mappings between the vocabularies of the intermediate ontology and each source
	Properties:	The output is equivalent to a relation and identifies related vocabulary entities (like ontology alignment)
Ontology Translation (first reading)	**Purpose:**	Translation to a different ontology representation language
	Input:	An ontology and a target ontology representation language
	Output:	An ontology expressed in the target language
	Properties:	Should produce an equivalent ontology, if possible
Ontology Translation (second reading)	**Purpose:**	Implementation of a vocabulary mapping
	Input:	An ontology and a mapping
	Output:	An ontology
	Properties:	Implements the vocabulary change to the source ontology as specified by the input mapping
Ontology Evolution	**Purpose:**	Respond to a change in the conceptualization or the domain
	Input:	An ontology and a (set of) change operation(s)
	Output:	An ontology
	Properties:	Implements a (set of) change(s) to the source ontology
Ontology Versioning	**Purpose:**	Transparent access to different versions of an ontology
	Input:	Different versions of an ontology
	Output:	A versioning system
	Properties:	Uses version ids to identify versions; provides transparent access to the "correct" version; determines compatibility
Ontology Integration	**Purpose:**	Fuse knowledge from ontologies covering similar domains
	Input:	Two ontologies (covering similar domains)
	Output:	An ontology
	Properties:	Fuses knowledge to cover a broader domain
Ontology Merging	**Purpose:**	Fuse knowledge from ontologies covering identical domains
	Input:	Two ontologies (covering identical domains)
	Output:	An ontology
	Properties:	Fuses knowledge to describe the domain more accurately

Table 2.1. Summary of the various subfields of ontology change

2.3. Description Logics (DLs) and Web Ontology Language (OWL)

Ontology Representation Formalisms

Several formalisms have been used for the representation of ontologies over the years, like frame-based and object models, Telos [MBJK90], SHOE [HH00], DAML+OIL [HHPSBL+01], RDF [MSB06] and others. Recently, DLs [BCMNPS02] and OWL [DSBHH+04] have emerged as the dominating formalisms for ontological representation. In this thesis, we will assume that an ontology is represented using some kind of DL; since the interesting members of the OWL family can be viewed as DLs [HPS04] this assumption is general enough to cover the most commonly used ontology representation formalisms.

Apart from their popularity and generality, DLs were chosen because their formal properties have been extensively studied in the literature, they have a clear formal semantics and enough diversity to cover very different representation needs; more arguments in favor of this choice can be found in later sections and in [BHS03]. In this section, we will make a brief presentation of DLs and OWL, outlining the features, properties and results that will be useful in our subsequent discussion. For a more detailed account on DLs, refer to [BCMNPS02]; for OWL, refer to [DSBHH+04].

Description Logics (DLs)

The term Description Logics (or DLs for short) refers to a family of knowledge representation languages, heavily used in the Semantic Web [BCMNPS02]. Most DLs can be viewed as fragments of FOL, but some of them go beyond standard FOL expressivity [Bor96]. DLs are descendants of semantic networks and frame-based knowledge representation models [BHS03].

Knowledge in DLs is structured in a formal, yet intuitive way. The basic "building blocks" used to represent knowledge in DLs are *classes*, *roles* and *individuals*. Classes represent concepts, i.e., groups of objects which share some common property; examples of classes are PENGUIN, MALE, HUMAN etc. Classes can also be viewed as unary relationships (unary predicates in the FOL terminology). Roles, on the other hand, represent certain relations between objects, such as HAS_CHILD, MARRIED_TO, LARGER_THAN etc. Roles are binary relationships (binary predicates in the FOL terminology). Finally, individuals represent the objects themselves, which are used to populate classes and roles; examples of individuals are BOB, JOHN, PENGO etc. Individuals can be viewed as constant symbols.

The allowable names for classes, roles and individuals form a (usually infinite) set, called the *namespace* of the DL. We will use the symbol N_{DL} to refer to the namespace of a DL; we will assume that this set is split into three disjoint sets namely N_{DL}^{C}, N_{DL}^{R}, N_{DL}^{I}, denoting the classes, roles and individuals of the DL namespace respectively. Obviously the following relations hold:

$$N_{DL} = N_{DL}^{C} \cup N_{DL}^{R} \cup N_{DL}^{I}$$

$$N_{DL}^{C} \cap N_{DL}^{R} = N_{DL}^{C} \cap N_{DL}^{I} = N_{DL}^{I} \cap N_{DL}^{R} = \varnothing$$

The elements of the namespace are combined to form DL formulas using *operators* and *connectives*. Operators allow the formation of more complex real-world constructions, called *terms*, expressible using more "primitive" elements (i.e., classes, roles and individuals). For example, we could represent the concept of a male human who has at least one human child, using the term:

$$\text{HUMAN} \sqcap \text{MALE} \sqcap \exists \text{HAS_CHILD.HUMAN}$$

The above term conceptualizes the concept of a father; if this concept is important for our conceptualization, we may wish to give it a separate class name, i.e., creating the class FATHER. To express the fact that FATHER is exactly the same concept as the one described above we would need an *axiom*; axioms are formed using some connective. In this particular case, the needed axiom is:

$$\text{FATHER} \equiv \text{HUMAN} \sqcap \text{MALE} \sqcap \exists \text{HAS_CHILD.HUMAN} \qquad (1)$$

Knowledge in DLs is represented using such axioms. Thus, a *Description Logic Knowledge Base (DL KB) is simply a set of DL axioms expressing knowledge regarding a domain of interest*.

There are several different operators that are allowed in a DL, each with its own syntax, arity and semantics (for example: $\neg$, $\sqcap$, $\sqcup$, $\exists$, $\forall$, $(\leq_n)$, $(\geq_n)$, $\{\ldots\}$, $^-$). There are also 0-ary operators (constants), like $\bot$, $\top$. Similarly, there are several different connectives, with different arity, semantics and syntax, like $\sqsubseteq$, $\equiv$, $\sqsubset$, $\not\sqsubseteq$, disj(.,.), Trans(.). The set of operators allowed in a given DL will be denoted by O_{DL}; the set of connectives will be denoted by C_{DL}.

Each operator can be viewed as a function; for example the '$\sqcap$' function maps each pair of concepts A, B to the concept A$\sqcap$B. We will use the standard notation in the literature for these functions (e.g., A$\sqcap$B, $\exists$R.A, etc) instead of the more formal prefix one (e.g., $\sqcap$(A,B), $\exists$(R,A), etc). Similarly, connectives are actually relations, but we will use the standard infix notation (e.g., A$\equiv$B, A$\sqsubseteq$B, etc) instead of the more formal prefix (e.g., $\equiv$(A,B), $\sqsubseteq$(A,B), etc) or mathematical one (e.g., (A,B)$\in\equiv$, (A,B)$\in\sqsubseteq$, etc).

One interesting feature of the family of DLs stems from the variety of the operators and/or connectives that can be defined: each particular DL allows for only a subset of all these proposed operators and connectives. This way, one can trade expressive power for computational efficiency and vice-versa, selecting the appropriate blend of connectives and operators for the application at hand. Some particular combinations have been considered more useful for practical applications, so they have been explicitly studied [BCMNPS02], but it is possible that new combinations (i.e., new DLs), as well as new operators and connectives will be proposed in the future.

Another factor that further increases this diversity is the variety of axiom types allowed in a DL. For example, axiom (1) can be viewed as a definition of the concept FATHER; in other words, the class name FATHER can be viewed as an abbreviation for HUMAN $\sqcap$ MALE $\sqcap$ $\exists$HAS_CHILD.HUMAN. Such axioms are called *definitorial*. But there are other axiom types as well:

$$\text{HUMAN} \sqsubseteq \forall \text{HAS_CHILD.HUMAN} \qquad (2)$$

$$\text{MARRIED_TO} \equiv \text{MARRIED_TO}^- \qquad (3)$$

$$\text{MARRIED_TO (BOB, MARY)} \qquad (4)$$

Axiom (2) has the form of an integrity constraint, stating that all humans' children must be humans as well. It is also *cyclic*, in the sense that the same class (HUMAN) is used in both the left and right hand side of the axiom. In some cases, no axiom is cyclic by itself, but a cycle occurs in a sequence of axioms; in this case we have a cyclic set of axioms. Certain DLs overrule cyclic axioms or cyclic sets of axioms; other DLs allow only for definitorial axioms, like (1). Axiom (3) is also cyclic and deals with roles; it states that the role MARRIED_TO is symmetric. There are DLs where roles are considered second-class objects, so no axioms

involving role terms are allowed. Axiom (4) deals with individuals, stating that the individual BOB is married to the individual MARY. Axioms containing assertions about individuals (also called *assertional axioms*) form the *Abox*; axioms dealing with concepts and roles only (also called *terminological axioms*) form the *Tbox*. In certain contexts, the Tbox is viewed as the schema of a DL KB, expressing certain facts and relations regarding the real world, while the Abox forms the data, i.e., the part of the KB dealing with the individual objects in the domain of interest.

In our context, the strict distinction between Tbox and Abox is not useful. In ontologies, schema and data are often intermingled in such a way that it is hard to distinguish where the ontology ends and its instances begin [NK04]. Furthermore, a distinction between these two ontological "parts" would jeopardize our attempt for a unifying approach to changes in ontologies, as well as our proposal related to a formalism that will bridge the fields of belief change and ontology evolution (see section 3.3). For these reasons, we will not use this distinction; in the rest of this dissertation, we will assume that an ontology (or DL KB) may (but need not) be populated with individuals and that both schema and data in an ontology are of equal value and use the same representation formalism and semantics.

Semantics of DLs

The strongest feature of DLs is that they are equipped with formal, logic-based, model-theoretic semantics, unlike their predecessors, i.e., semantic networks and frames [BHS03]. The formal semantics of DLs is defined using *interpretations* $I=(\Delta^I,\cdot^I)$ which consist of a non-empty set Δ^I and a function $\cdot^I$ which maps every concept $A\in N_{DL}{}^C$ to a set $A^I\subseteq\Delta^I$, every role $R\in N_{DL}{}^R$ to a relation $R^I\subseteq\Delta^I\times\Delta^I$ and every individual $x\in N_{DL}{}^I$ to an object $x^I\in\Delta^I$. This mapping is extended to terms using the semantics of each operator; for example, $(A\sqcap B)^I=A^I\cap B^I$. The semantics of each connective determines whether an axiom is *satisfied* by a given interpretation; for example, the axiom $A\sqcap B\sqsubseteq C$ is satisfied by the interpretation I iff $(A\sqcap B)^I\subseteq C^I$ which is equivalent to $A^I\cap B^I\subseteq C^I$ by the semantics of $\sqcap$. The semantics of each connective and operator is usually straightforward; the interested reader is referred to [BCMNPS02] for a full account.

These semantics can be easily generalized to sets of axioms: a set of axioms is *satisfied* by an interpretation I iff all the axioms in the set are satisfied by I. A set of axioms P *implies* a set of axioms Q (denoted by $P\vdash Q$) iff all interpretations that satisfy P also satisfy Q. A set of axioms P is *equivalent* to a set of axioms Q (denoted by $P\cong Q$) iff $P\vdash Q$ and $Q\vdash P$. A set of axioms is *inconsistent* iff there is no interpretation satisfying it. Notice that inconsistent sets of axioms trivially imply all axioms; for this reason, inconsistent axioms (and sets of axioms) are often considered to carry no knowledge, as no meaningful inference can be made from them. For the same reason, inconsistent DL KBs are highly undesirable, so their study is very important for our purposes: any ontology evolution algorithm should first and foremost guarantee that the result of the change is not an inconsistent set of DL axioms.

The semantics described above allows us to conclude certain facts from any given DL KB. For example, the combination of axioms (3) and (4) above should allow us to conclude that the individual MARY is married to the individual BOB, i.e., the axiom:

MARRIED_TO (MARY, BOB) (5)

This intuitively expected result should be (and is) deducible from DL semantics in a formal manner. Such kind of inference capabilities are provided by DLs and are necessary for any kind of formal treatment of DL KBs. Other, closely related, types of inference services are

possible as well; these are irrelevant for our purposes, so we refer the reader to [BCMNPS02] for a full account. There has been a considerable amount of work dealing with the issue of reasoning in DLs, resulting in several algorithms and complexity results for various types of DLs (see, for example [BS01], [Cal96], [DLNS96], [HS01], [HST99], [HS00], [LS00], [SSS91]); a useful collection of such results can be found in [Zol06].

The formal semantics accompanying DLs has also allowed the development of several formal results relating DLs with other formalisms, such as modal logics [BCMNPS02], logic programs and rules [GHVD03] or (fragments of) FOL [BCMNPS02], [Bor96]. It has also allowed the development of certain DL extensions, like temporal DLs [AFMWZ01], nonmonotonic DLs [DNR97] or disjunctive DLs [MLB05].

Web Ontology Language (OWL)

A formalism closely related to DLs is the Web Ontology Language [DSBHH$^+$04], known as OWL; OWL is expected to play an important role in the future of the Semantic Web, as it has become a W3C Recommendation. OWL is actually a family of knowledge representation languages for ontologies: it comes in three "flavors" (or "species"), namely OWL Full, OWL DL and OWL Lite, with varying degree of expressive power and reasoning complexity. In OWL, knowledge is represented using an RDF-like [MSB06] syntax. OWL contains several features allowing the representation of complex relationships between classes, roles and individuals in a pattern very similar to the one used in DLs; this close relationship was verified in [HPS04], where OWL DL and OWL Lite, with some secondary (annotation) features removed, were shown equivalent to the DLs $SHOIN^+(D)$ and $SHIF^+(D)$ respectively. On the other hand, OWL Full provides a more complete integration with RDF, containing features not normally allowed in DLs (see [HPS03] for the main semantical differences between RDF and logic-based languages, like FOL and standard DLs); furthermore, its inference problem is undecidable [HPS04]. OWL was influenced by several different formalisms that are used for ontological representation, like DLs [BCMNPS02], RDF [MSB06], DAML+OIL [HHPSBL$^+$01] and the frames paradigm; OWL comprises an effort to develop an ontological representation language that accommodates its various influences in one widely accepted formalism [HPSH03].

Ontologies and DLs

As already stated, we will assume that ontologies are DL KBs under some kind of DL. This assumption prohibits the use of some ontological features which are available in OWL (for example), but not available in standard DLs; these include certain meta-logical annotation features stating the author, version etc of the ontology or the feature of including an ontology in another (ontology reuse). However, these restrictions are not important for our purposes; as we will explain later, our focus is on determining the modifications that must be made upon an ontology in response to a certain need (phases two and three of ontology evolution [SMMS02]). Updating meta-logical features, like the author for example, is a trivial operation in this respect, even though certain types of change (like the change of the version number for example) may induce complications in other phases of the ontology change process (in this case, ontology versioning). Regarding the ontology reuse feature, we will assume that there is a preprocessing phase in which the included ontology (or ontologies) is (are) physically copied to the including ontology; for our purposes, this is the same as keeping them separate, even though the two approaches are not generally equivalent [FP05]. For more details on the

impact of annotation features on our approach, see the part of section 5.5 that deals with the annotation features of OWL.

Another thorny issue is posed by the signature structure of the ontology. The signature contains the elements that are relevant for the conceptualization of the ontology. Unfortunately, this information cannot be directly encoded using standard DL axioms. For more details on this issue, as well as for the proposed solution, refer to section 3.3.

Unless otherwise specified, we will not assume any special properties for the underlying DL, nor any specific operators and/or connectives; in effect, we will assume that ontologies are represented in some generic, unspecified DL. We will not even assume that the DL at hand is one of the DLs that have been studied in the literature; this allows our results to be useful in a variety of current and/or future DLs.

2.4. Belief Change

Introduction

The rich literature on belief change will be the main means towards our effort to develop an automatic, rational and effective ontology evolution algorithm. In this section we perform a brief survey on the belief change literature, thus providing the necessary introduction to the main concepts and ideas expressed in the field and making this dissertation self-contained. Despite the formal character of most works on belief change, our survey will be kept at a rather informal level for presentation purposes; we will get into the formal details only in cases where such details are relevant to our subsequent discussion.

Basic Definitions

One's beliefs are not generally static, but evolve over time. This may be because of several reasons: new, previously unknown, classified, or otherwise unavailable information may have become known; a new observation or experiment may reveal a new fact; or the actual world (domain of interest) may change, as domains are in general not static themselves. In all these cases, the beliefs held should be somehow adapted to the new information. Alternatively, the new information could be ignored.

These thoughts are valid for any structure (i.e., KB) holding information (beliefs, facts, rules, data etc) regarding a domain of interest and they have found applications in diverse areas like learning [Kel98], software engineering [Wil98] and marketing research [Wil98]. The research area of *belief change* deals with *the adaptation of a KB to new information* [Gar92a]. While knowledge representation deals with static knowledge and how this can be represented, belief change deals with the dynamic aspects of knowledge, i.e., how this knowledge evolves.

Representation Issues

Several philosophical and practical issues related to belief change can be identified. A detailed description of such issues is outside the scope of this work; we will briefly touch the main tradeoffs and problems that have been dealt with in the field and point to certain references where the various issues are analyzed in more detail. The purpose of this brief review is to show the diversity of the problem, as well as the variety of approaches that have been considered. The amount of diverse work performed in the area, as well as the maturity of

many approaches, allow us to hope that several belief change techniques, ideas or intuitions will prove useful in the field of ontology evolution.

Before any practical approach on the subject of knowledge changing, one should first decide on the formalism(s) that will be considered for the representation of knowledge. Most approaches in the literature deal with a family of logics which satisfies certain quite general assumptions; this family includes most classical logics like Propositional Logic (PL) or First-Order Logic (FOL) [Bur98], [End72], [Men87]. Examples of such works are: [AGM85], [AM85], [AM86], [Boch02a], [Boch02b], [Del03], [GM88], [Gro88], [Mak85], [Neb89]. Other belief change approaches are restricted to the most studied logic in the area, namely PL: [Dal88a], [Dal88b], [FP01a], [FP01b], [FP03], [KM90b], [Web86]. There are few exceptions that deal with non-classical logics, most notably [BAGM99], dealing with nonmonotonic theories and [FUV83], which focuses on the database paradigm. To the best of the author's knowledge, there is no belief change approach dealing with DLs or related languages; this shows that the issue of ontology evolution has not been addressed by the belief change community. However, there are some recent works (at a preliminary stage) which try to apply existing belief change approaches to DLs ([KL04], [LM04], [MLB05]), using intuitions similar to the ones described in this thesis (see section 3.5 for details), as well as some which apply belief change techniques to the evolution of concepts [Foo95], [Wass98], [WF99].

The selection of the underlying language is not the only issue involved; more fundamental questions related to the representation of knowledge should be resolved even before deciding which logic to use. For example, should we consider our knowledge being represented in the form of a *belief base* or a *belief set*? Belief bases are small, finite sets of expressions of the underlying language; belief sets are large, infinite structures, closed under logical consequence. In other words, belief sets contain explicitly all the knowledge deducible from the KB, while belief bases contain some of the information explicitly, the rest being deducible via the standard reasoning (inference) mechanism of the underlying logic.

As far as knowledge representation is concerned, the two views are the same, as we can calculate the logical consequences of a belief base in order to reveal the full knowledge of a KB, whenever necessary. But in belief change things are different. When changes are performed upon a belief base, we temporarily have to ignore the logical consequences of the base; in effect, there is a clear distinction between knowledge stored explicitly (which can be changed directly) and knowledge stored implicitly (which cannot be changed, but is indirectly affected by the changes in the explicit knowledge). In the belief set approach, all the information is stored explicitly, so there is no distinction between implicit and explicit knowledge. In the belief base approach, our options for the change are limited to the base itself, so the changes are generally more coarse-grained; this limits our options for "proper" belief change. On the other hand, belief sets are infinite structures, so it is not possible to deal with them directly in practical applications, so, in practice, only a small subset of a belief set is explicitly stored; in contrast to the belief base approach however, the implicit part of our knowledge is assumed to be of equal value to the explicitly stored one. Some thoughts on the connection between the two approaches can be found in [Fuhr91], [Han93]; in [Han96] a thorough knowledge-level discussion and motivation on belief base operations is presented; a comparison of the approaches can be found in [Han92], [Han98].

A related philosophical consideration studies how our knowledge *should* be represented. There are two viewpoints here: *foundational* theories and *coherence* theories. Under the foundational viewpoint, each piece of our knowledge serves as a justification for other beliefs;

our knowledge is like a pyramid, in which "every belief rests on stable and secure foundations whose identity and security does not derive from the upper stories or sections" [Sosa80]. This viewpoint is closer to the belief base approach. On the other hand, under the coherence theory, no justification is required for our beliefs; a belief is justified by how well it fits with the rest of the beliefs, in forming a coherent and mutually supporting set; thus, every piece of knowledge helps directly or indirectly to support any other. In this sense, knowledge is like a raft, "every plank of which helps directly or indirectly to keep all the others in place, and no plank of which would retain its status with no help from the others" [Sosa80]. Coherence theories match with the use of belief sets as a proper knowledge representation format. A more detailed study on these issues can be found in [Gar92b].

Belief Change Operations

Another issue deals with the nature of the new information; in most cases, the new information is something that must be *added* to our knowledge. In other cases, the new information represents something which is wrong with our beliefs and should be *retracted*. In [AGM85] three different types of belief change were identified. The simplest one is *expansion*, which refers to the reckless addition of information to our knowledge, without taking any special provisions to ensure the quality (i.e., consistency) of the KB after the addition. This is not an interesting operation, because, if the new information contradicts the currently held beliefs then the result will be an inconsistent (and thus useless) KB. Furthermore, it is trivial to perform an expansion operation: the result of an expansion is simply the union of the KB and the change.

The second operation is *revision*, which is the most useful operation in terms of practical applicability. Revision is similar to expansion, with the important difference that the result should be a consistent set of beliefs. In order to achieve this, one may, in some cases, need to abandon some of the currently held beliefs while adding the new information. The tough part of revision is how to select the beliefs that should be abandoned.

Contraction is, according to [Gar92a], the most fundamental operation and, consequently, the most important operation for theoretical purposes. Contraction is used when one wishes to consistently remove information from a KB. This may be necessary when a piece of information becomes unreliable and we would like to stop believing it. A contraction operation should remove the unreliable information from both explicit and implicit knowledge; simply retracting the information from explicit knowledge may not be enough, as it may re-emerge as a consequence of the remaining beliefs. Thus, a contraction operator may be forced to also remove beliefs which on first look seem unrelated to the retracted piece of knowledge.

The above operations deal with a static world; in effect, the world itself does not change, but our perception of the world does. Thus, revision and contraction are used when some new information about the real world has been disclosed, forcing us to change our conceptualization of the world in order to represent it in a (hopefully) more accurate manner. But this is not the only change possible, because the real world might change as well; in this case, the KB should be adapted to the new reality. The semantics of this kind of change is quite different, forcing us to introduce a new pair of operations, namely *update* and *erasure* [KM90a]. Update is similar to revision (it refers to addition of information) while erasure is similar to contraction (it refers to removal of information); however, they both apply when the

world dynamically changes, which makes them substantially different from their "static" counterparts.

One example may help clarify the situation: suppose that we have three independent switches S_1, S_2 and S_3 and the information that exactly one of S_1, S_2 or S_3 is on. Our knowledge could be represented by the PL KB:

$$(S_1 \wedge \neg S_2 \wedge \neg S_3) \vee (\neg S_1 \wedge S_2 \wedge \neg S_3) \vee (\neg S_1 \wedge \neg S_2 \wedge S_3)$$

Now, suppose that, through observation, we notice that S_1 is on. The proper reaction to this kind of information is to assume that S_2 and S_3 are off, as this coincides with both the fact that exactly one switch is on (old KB) and with the fact that S_1 is on (change). So the new KB should be:

$$S_1 \wedge \neg S_2 \wedge \neg S_3$$

In this particular case, we choose, among the three possible states of the world, the one that agrees with the new information. This is the case in a revision operation.

Things are very different when the world dynamically changes. Suppose that we send a robot into the room with the order to turn S_1 on; then, we have no reason to assume that S_1 was on before the robot was sent in. The robot changes the real world by switching S_1 on, so S_1 should be on in the new KB, but that's all we know. So we should assume that at the end of the robot's actions, either S_1 alone, or S_1 and S_2, or S_1 and S_3 are on. The new KB should be:

$$(S_1 \wedge \neg S_2 \wedge \neg S_3) \vee (S_1 \wedge S_2 \wedge \neg S_3) \vee (S_1 \wedge \neg S_2 \wedge S_3)$$

We conclude that the expected result is different than in the case of revision; we don't choose among the possible worlds, rather we alter each of the possible worlds independently so as to coincide with the new fact. This is the case in an update operation.

Notice that in our example, the original KB and the information that initiated the change (S_1) are formally represented in the same way in the two operations; however, the desired result of the change is different. For this reason, the introduction of the new operations is necessary in order to capture the different semantics that a change can take. For a thorough discussion on the differences between the various operators, see [KM90a]; for an approach that handles sequences of different operations see [HD05].

Semantics for the Change

The format and semantics of the new information may also vary. Under the standard approach, the "new information" is a single expression of the underlying logic, which is received and accommodated to our knowledge in an autonomous, independent manner. In other approaches, the new information can be a whole set of expressions and may have *package* or *choice* semantics [FH94]. This semantics was introduced for contraction: under package semantics, all expressions in the set must be retracted from the KB; under choice semantics, it is sufficient that at least one of them is retracted. In other cases, belief change is considered to be a continuous process, so each change is related to previous (and future) changes. Under this viewpoint, the change is not a one-off, standalone process, but a set of more elementary changes occurring in a sequence. This type of belief change has different semantics and is treated in the field of *iterated belief change* [Boch02b], [DP97], [HD05], [Lib97], [Nay94]. In other cases, the new information may be a whole new KB; in this case, we are dealing with *belief merging* [Kon00], [KPP05] which is the analogous of ontology merging for KBs.

Postulation Versus Explicit Construction

Once the aforementioned technicalities have been dealt with, one has to decide how the change itself should be performed. In general, there are two approaches here: *postulation* and *explicit construction* [Mak85]. Under the postulation approach one seeks to formulate a set of formal conditions (usually in the form of postulates) that a belief change algorithm should satisfy in the given context. Under the explicit construction approach, one seeks explicit algorithms or constructions leading to algorithms, which are suitable for the given context.

The two approaches are not rivalrous but complementary [Mak85]. In the process of developing an explicit construction, one may identify certain desired conditions for this construction (or algorithm), something like a "request specification", that could lead to a postulation. On the other hand, postulation may produce better results if its initial findings are checked with an explicit construction that satisfies (or does not satisfy) the postulates; this construction may help in determining the applicability and rationality of the proposed postulation and lead to refinements and improvements.

Both methods have been used in belief change and both have given interesting results. Explicit belief change algorithms are presented in [Dal88a], [Dal88b], [Del03], [Doy79], [Web86]; constructions that lead to families of algorithms are given in [AGM85], [AM85], [AM86], [Boch02a], [Boch02b], [Fuhr91], [GM88], [Gro88], [HD05], [KM90b], [Neb89], [Wil97]; possible postulations of the process are provided in [AGM85], [Del03], [Doy91], [GN94], [Han96], [HD05], [KM90a], [KM90b]. Such works have been developed in a parallel and complementary manner [Mak85]; this is also apparent by the overlap in the above lists of works. On the other hand, current research on ontology evolution uses only the explicit construction method; one interesting side-effect of our approach is that it allows the development of a postulation method for this problem.

Issues Involved in the Change Process

There are several philosophical questions related to belief change for which, in general, there is no unique answer because such an answer depends on the application at hand. We have already seen several such issues, mostly related to the representation of the change and knowledge; in the following, we will briefly discuss those related to the determination and implementation of the actual change, which is the main focus of belief change. The properties of any given approach to belief change are determined by the stance it takes towards these concerns; this makes each work more suitable for a certain context and class of applications.

One such issue is related to the acceptance of the new information. It is usually assumed that the new information (revision, contraction or other) is accepted unconditionally, so the resulting KB should contain the new information (or should not contain it, depending on the operation at hand). This implies a complete reliance to the incoming data, according to the *Principle of Primacy of New Information* [Dal88a]. This principle coincides with common intuition, because the new information generally reflects a newer and more accurate view of the domain. Alternatively, one could review the new data under a critical eye and possibly reject it, partially or totally; this is the case in the so-called *non-prioritized belief change* [Han99a]. This is most useful in an agent communication context, or when the information may be obtained by unreliable or untrustworthy sources.

Some researchers argue that the result of a change should not be affected by the syntactical representation of the KB or the change (*Principle of Irrelevance of Syntax* [Dal88a]). Normally, the principle applies in both the KB and the change; however, it is generally

possible that it could be applied in the KB only or in the change only. This principle generally fails for belief bases (under the foundational viewpoint), because logically equivalent bases may be formed using completely different sets of axioms, implying different justifications and, thus, different ways of adapting the KB to the new data [Han96]. A stronger version of this principle may be of use in the foundational case. For belief sets (and coherence theories) though, this principle is generally valid [AGM85], [Dal88a], [Gar92a], because the intuitively expected result of a change is determined by the logical force (semantics) of the operands (KB and change), rather than their syntactical formulation.

As already argued, inconsistent KBs (under classical logic) carry no interesting information and, consequently, should be avoided. For this reason, it is generally accepted that the result of a change should be a consistent KB, according to the *Principle of Consistency Maintenance* [Dal88a]. This is not an issue for nonmonotonic and paraconsistent KBs, in which the underlying logic itself has an internal mechanism to deal with inconsistencies [Ant97], but for classical logic the principle is valid (see also [KLM05] for a recently proposed alternative approach). The only thing that remains to be settled is the exact meaning of the term "consistency"; in the belief change literature, the meaning of the word is clear: an inconsistent KB is one which implies a proposition that is tautologically false. For ontology evolution however, the term "consistency" has been used (others would say abused) to denote several different things; we will address this issue at a later section.

Two more basic principles have been proposed in the literature: the *Principle of Fairness* guarantees determinism and reproducibility of the result of a change, while the *Principle of Adequacy of Representation* guarantees that the new (modified) KB will be represented using the same underlying formalism as the old one [Dal88a]. These principles are intuitively obvious, as well as necessary for technical reasons, so they are often implicitly used in the literature without explicit introduction (but see [Doy91] for some arguments against determinism).

Undoubtedly, the most important principle related to the way a change is implemented is the *Principle of Minimal Change* [KM90b], which can be found by several names in the literature, like the *Principle of Persistence of Prior Knowledge* [Dal88a] or the *Principle of Conservation* [Gar92b]. This principle states that the new KB should be as "close" as possible to the original KB; in other words, from all the possible change results that satisfy the other principles, one should choose the one that retains "the most information" from the old KB.

To the author's knowledge, there is little debate on whether this principle should be valid for belief change operations; this is taken for granted in most works (but see [Doy91] for an exception). However, there is no consensus on how this principle should be formally expressed. For such a formulation, the exact meaning of "closeness of KBs" needs to be defined, as well as how the "loss of information" can be measured. There have been several proposals on metrics that count information loss in several ways, being used in different algorithms or representation results (for example [AM85], [Dal88a], [Dal88b], [Del01], [Fuhr91], [GM88], [Gro88], [KM90b], [Mak85], [Web86]); there are also different postulations that capture this principle in a different manner [AGM85], [Han96] and long debates on the pros and cons of each postulation [Gar92a], [Han92], [Mak87].

One of the reasons for this diversity is that logical considerations alone do not tell us what has to be given up in a certain change [Gar92a]; extra-logical information needs to be considered when it comes to choosing. The way this information is structured and used

determines the stance of the approach towards the Principle of Minimal Change. In fact, the formulation of this principle is in the core of each belief change algorithm or postulation attempt, being the very essence of belief change and determining the properties of any given approach to a large extent. This notion also provides interesting connections between belief change and several related research areas, like defeasible inference, counterfactual conditionals and others [Mak93].

Discussion

The aforementioned considerations form only a partial list of the issues that have been addressed in the belief change literature. Our analysis shows that the determination of the change(s) to be made in response to some new data is a complex and multifaceted issue and that several considerations need to be taken into account before choosing the modifications to be performed upon a KB. The same considerations hold for any type of knowledge change, including ontology evolution. For this reason, we believe that current ontology evolution approaches handle the problem in an inadequate manner, as they are not even considering most of these issues in the algorithms' determination.

In this respect, it is interesting to note that, in the belief change literature, there is no human involved in the process of change, unlike the ontology evolution literature. All the approaches in belief change deal with the problem in a fully automatic manner. In fact, to the author's knowledge, the option of using a human in the loop of belief change was never even considered as an option, despite the complexity of the problem. This fact forms an additional argument in favor of the use of belief change techniques in ontology evolution, as such techniques will arguably lead to automatic methods for dealing with this problem too.

2.5. AGM Theory

Introduction

In this section, we will briefly present the AGM theory [AGM85]; this theory is presented in more detail than the other belief change approaches because of its special importance for both the belief change community and for our study in this thesis. Before going into the details of the AGM theory, we will present one very important and very general logical framework, originally introduced by Tarski in 1928, upon which both the AGM theory and our approach are based.

Tarski's Logical Framework

Tarski's logical framework defines a *logic as a pair* $<L,Cn>$, where L is the set of expressions (propositions) of the underlying language and Cn is a function mapping sets of propositions to sets of propositions (*consequence operation*). The intuitive meaning of Cn is that a set X implies exactly the propositions contained in Cn(X). In other words, L represents the language (i.e., what is expressible in the logic), while Cn determines what is implied by each proposition (and set of propositions).

The basic block of knowledge in this framework is the *belief*, which is simply defined as a set of propositions of the underlying language L (i.e., a subset of L). The inference mechanism of the logic is constrained by placing certain intuitive restrictions on the Cn operator that force it to behave in a rational manner. More specifically, Cn was constrained to satisfy the following (intuitively obvious) conditions:

For all $X \subseteq L$, $X \subseteq Cn(X)$ (*inclusion*)
For all $X \subseteq L$, $Cn(X)=Cn(Cn(X))$ (*iteration*)
For all $X, Y \subseteq L$, if $X \subseteq Y$ then $Cn(X) \subseteq Cn(Y)$ (*monotony*)

Using this operator one can easily define an *inference relation* between sets (beliefs) as follows:

$$X \vdash Y \Leftrightarrow Y \subseteq Cn(X)$$

It can be shown that the inference relation thus defined satisfies the following properties:

For all $X, Y \subseteq L$, if $X \subseteq Y$, then $Y \vdash X$ (*reflexivity*)
For all $X, Y, Z \subseteq L$, if $X \vdash Y$ and $X \cup Y \vdash Z$, then $X \vdash Z$ (*transitivity*)
For all $X, Y, Z \subseteq L$, if $X \vdash Y$, then $X \cup Z \vdash Y$ (*weakening*)

The opposite can also be shown; given a relation $\vdash$ which satisfies these properties (i.e., reflexivity, transitivity and weakening), the function:

$$Cn(X)=\{y \in L \mid X \vdash \{y\}\}$$

is a consequence operation (i.e., it satisfies inclusion, iteration and monotony). Thus, there is a tight correspondence between consequence operations and inference relations, making them interdefinable. This way, a logic can also be defined as a pair $<L,\vdash>$ (see also [Fuhr91], [Mak85]).

In Tarski's framework, a belief X *is implied by* a belief Y iff $X \subseteq Cn(Y)$, or, equivalently, iff $Y \vdash X$. Two beliefs are called *equivalent* (denoted by $X \cong Y$) iff $X \vdash Y$ and $Y \vdash X$, or, equivalently, iff $Cn(X)=Cn(Y)$. We will use the notation [X] to denote the class of beliefs which are equivalent to X.

Notice that Tarski's consequence operator has the properties of an *algebraic closure* operator [BS81] (or an *abstract mathematical closure* operator [Wiki06]); therefore, calculating the consequences of a set corresponds to taking the closure of this set. For this reason, we will often use the term *closure* to denote the process of calculating the consequences of a set X (i.e., calculating Cn(X)).

Several results can be shown using these simple properties alone. The ones that will be most useful in this dissertation have been collected in the following lemma:

Lemma 2.1. Consider a logic $<L,Cn>$, under Tarski's model. The following hold:

- $Cn(Cn(X) \cup Cn(Y))=Cn(Cn(X) \cup Y)=Cn(X \cup Y)$
- $Cn(Cn(X) \cap Cn(Y))=Cn(X) \cap Cn(Y)$
- If $X \subseteq Cn(Y)$ then $Cn(X \cup Y)=Cn(Y)$
- If $X \subseteq Cn(Y)$ then $Cn(X) \subseteq Cn(Y)$
- $Cn(X \cap Y) \subseteq Cn(X) \cap Cn(Y) \subseteq Cn(Cn(X) \cap Cn(Y))$
- $Cn(X \cup Y) \supseteq Cn(X) \cup Cn(Y)$

Proof

The proof uses only the iteration, inclusion and monotony properties of Cn. It is easy and is omitted. □

As we will see at a later point, Tarski's general framework engulfs most known knowledge representation languages, including (monotonic) DLs and OWL, which are the focus of our work, as well as the logics considered by most belief change approaches. Thus, it provides a common ground for interaction between ontology representation languages and belief change algorithms; this fact makes Tarski's framework very important for our purposes.

AGM Assumptions and Operations

The AGM theory [AGM85], developed by Alchourron, Gärdenfors and Makinson (AGM for short), is undoubtedly the most influential approach in the field of belief change. AGM attempted to formalize the field of belief change by addressing the general problem of finding the properties that a rational belief change operator should satisfy. In effect, they used the postulation method, as opposed to an explicit construction which was the general trend at the time. They dealt with two important operations (revision, contraction) and one trivial one (expansion), providing one set of postulates (the *AGM postulates*) for each of revision and contraction.

AGM made several, quite general assumptions when formulating their theory. As already mentioned, they followed Tarski's viewpoint on the definition of a logic, so a logic, in the AGM theory, is a pair <L,Cn>. This assumption provides the link between their theory and the world of ontologies, as ontological representation languages (like DLs and OWL) can also be viewed as <L,Cn> pairs, as mentioned before. However, AGM made additional assumptions as well: they assumed that the underlying logic is closed under the standard operators of PL ($\neg$, $\wedge$, $\vee$, $\rightarrow$ etc) and that the consequence operator includes classical tautological implication, is compact and satisfies the rule of introduction of disjunctions in the premises (hereafter referred to as the IDP rule). The formal expression of this rule is:

$$Cn(X\cup(Cn(Y)\cap Cn(Z)))=Cn(X\cup Y)\cap Cn(X\cup Z), \text{ for all } X, Y, Z\subseteq L \qquad \text{(IDP rule)}$$

Regarding the operations of contraction and revision, AGM assumed that a KB is a set of propositions of the underlying logic (say $K\subseteq L$) which is closed under logical consequence (i.e., $K=Cn(K)$), also called a *theory*. This means that AGM use belief sets, i.e., they adopt the coherence model. The KB can be contracted or revised with any single expression $x\in L$ of the underlying logic. Thus, the operations of contraction and revision can be formalized as functions mapping the pair (K, x) to a new KB K' (denoted by $K'=K-x$, $K'=K+x$ respectively). These assumptions allow any binary operator to be a "contraction" or "revision" operator, which, of course, should not be the case; for this reason, AGM introduced several restrictions (in the form of rationality postulates) on the result of such operators.

The AGM Postulates

Regarding contraction, AGM restricted the result to be a theory itself (see postulate (K–1) below). As already stated, contraction is an operation that is used to remove knowledge from a KB; thus, the result should not contain any new, previously unknown, information (K–2). Removal of information should occur only when necessary (K–3). Moreover, contraction is supposed to return a new KB such that the contracted expression is no longer believed or implied (K–4). Finally, the result should be syntax-independent (K–5) and should remove as little information from the KB as possible, in accordance with the Principle of Minimal Change (K–6). These intuitions were formalized in the *basic AGM postulates for contraction*, which are the following [AGM85]:

(K–1)	$K-x$ is a theory	(*closure*)
(K–2)	$K-x\subseteq K$	(*inclusion*)
(K–3)	If $x\notin Cn(K)$, then $K-x=K$	(*vacuity*)
(K–4)	If $x\notin Cn(\emptyset)$, then $x\notin Cn(K-x)$	(*success*)
(K–5)	If $Cn(\{x\})=Cn(\{y\})$, then $K-x=K-y$	(*preservation*)
(K–6)	$K\subseteq Cn((K-x)\cup\{x\})$	(*recovery*)

Likewise, for revision, the result should be a theory (K+1); the new information should be included in the result, because we are dealing with revision (K+2). The result should be consistent (K+4) and should be independent of the exact syntactical formulation of the revision (K+5). If the new information contradicts the KB, then some information should be removed to avoid contradictions; but information should be removed only when necessary, and only as little as possible to guarantee the consistency of the result (see (K+3) and (K+6) below). The *basic AGM postulates for revision* are the following [AGM85]:

(K+1) K+x is a theory
(K+2) $x \in K+x$
(K+3) If $\neg x \notin Cn(K)$, then $K+x=Cn(K \cup \{x\})$
(K+4) If $\neg x \notin Cn(\varnothing)$, then K+x is consistent
(K+5) If $Cn(\{x\})=Cn(\{y\})$, then $K+x=K+y$
(K+6) $(K+x) \cap K = K-(\neg x)$

In addition to these basic postulates, AGM proposed two, quite powerful, *supplementary* postulates [AGM85] for each operation, which concern composite belief change. The supplementary postulates for contraction are:

(K−7) $(K-x) \cap (K-y) \subseteq K-(x \wedge y)$
(K−8) $K-(x \wedge y) \subseteq K-x$, provided that $x \notin K-(x \wedge y)$

The supplementary postulates for revision are:

(K+7) $K+(x \wedge y) \subseteq Cn((K+x) \cup y)$
(K+8) $Cn((K+x) \cup \{y\}) \subseteq K+(x \wedge y)$, provided that $\neg y \notin K+x$

These postulates formally express the AGM stance on some of the philosophical debates that we described in the previous section, in effect describing the AGM notion of "rationality" for a contraction and revision operator. It must be emphasized that there is a whole class of operators that satisfies the AGM postulates. In fact, this is the role of postulation [Gar92a]: further refinement of such operators should be the subject of explicit constructions and more applied research. The AGM postulates determined a class of rational operators; the issue of selecting one of them for use in a particular application is greatly depending on the application itself, among other things.

Criticisms and Importance

The AGM model is still one of the dominating models in belief change research. The key reason for its importance lies in the wide acceptance that it received by fellow researchers, who agreed that the postulates express common intuition regarding the operations of revision and contraction. This is true because the AGM approach is based on simple arguments that reflect common intuition on what the effects of a revision or contraction operation should be.

The only postulate that has been seriously debated is the contraction postulate of recovery (K−6), as some works [Fuhr91], [Han96] state that (K−6) is counter-intuitive, while others state that it forces a contraction operator to remove too little information from the KB [Han99b]. However, it is generally acceptable that the recovery postulate cannot be dropped unless replaced by some other constraint that would somehow express the Principle of Minimal Change. Recovery is one possible formulation of this principle because it forces the contraction operator to remove only beliefs which are relevant to the contracted expression; if only such beliefs are removed, the re-introduction of the contracted expression should restore

the original KB to its full logical power. For a more thorough discussion on this postulate refer to [Mak87].

Despite these objections, the AGM postulates greatly influenced future postulation attempts as well as explicit constructions. It would not be an overstatement to say that this work set the foundations for future research on the field of belief change providing a theoretical basis upon which deeper results could be developed. Most subsequent works praised, criticized, commented, used, extended or provided alternative characterizations of the AGM model making it the most influential approach on belief change.

Results Based on the AGM Theory

The line of research that followed the publication of the AGM theory gave several interesting results. For example, certain explicit constructions for families of contraction or revision operators (like partial meet selection functions [AGM85], epistemic entrenchment [GM88], safe contraction [AM85] and systems of spheres [Gro88]) have been shown to produce exactly the class of operators that satisfies the AGM postulates for contraction and/or revision. The fact that these intuitively appealing constructions produce exactly the same operators as the AGM postulation approach does, forms an additional argument in favor of the importance of the AGM postulates as a determinant for rational revision and contraction operators.

Other results related to the AGM theory deal with the connection of the AGM postulates with other conditions or informal notions that have appeared in the literature; this study was made by AGM themselves in [AGM85], [Gar92a], [Mak85]. In [Wit02], the connection of fuzzy set theory [Zad65] with the AGM theory (and postulates) was studied. Alternative formulations of the postulates in the propositional setting were proposed in [KM90b]. Several researchers studied the belief change operations they proposed with respect to the AGM postulates [Dal88a], [Neb89]. Other researchers studied the postulates' effects and properties [AM86], [Mak85], [Mak87] or criticized the AGM model [Fuhr91], [Han96], [Han99b], sometimes providing alternative formalizations [Fuhr91], [Han96]. All these works show the influence of the AGM theory in subsequent belief change research.

A detailed account of all such results is outside the scope of this brief survey; however, we will describe one particularly interesting result, which is related to the connection between the operations of revision and contraction. This connection is described by the Levi and Harper identities [Gar92a]. The Levi identity has the following formulation:

$$K+x=Cn((K-\neg x)\cup\{x\})$$

The intuitive idea behind this identity is that, in order to revise a KB K with x, one needs to remove any possible source of contradiction (by contracting with $\neg x$) and then expand the result with x. This viewpoint makes revision a two-stage process: initially everything that implies $\neg x$ is removed from the KB; then the KB is expanded with x. The first step guarantees that no inconsistencies will arise during the second step. This identity allows us to produce a revision operator from a given contraction operator.

The Harper identity is actually equivalent to the AGM postulate (K+6). Its formulation is given from (K+6) by replacing x with $\neg x$ and vice-versa:

$$K-x=(K+\neg x)\cap K$$

The intuitive idea is the following: if we revise a KB K with $\neg x$, the revision operator will make sure that $\neg x$ is a consequence of the new KB; since the new KB is consistent it cannot

imply both x and ¬x at the same time. Therefore, the revision by ¬x will, as a side-effect, eliminate from the original KB any occurrences of x, as well as any propositions that imply x. If we keep from K+¬x the part that originally belonged to K, we will get a part of K that does not imply x. This process allows us to get a contraction operator from a revision operator.

These identities show that the two fundamental belief change operations are interdefinable. This importance of this fact is emphasized by their relation with the AGM theory: if a contraction operator satisfies the basic AGM postulates for contraction, then the revision operator that the Levi identity generates satisfies the basic AGM postulates for revision. The dual result holds for the Harper identity as well and similar results can be shown for the supplementary postulates. Therefore, the problem of defining a rational (in the AGM sense) revision operator is actually a different facet of the problem of defining a rational contraction operator and vice-versa.

Chapter 3: Belief Change in Ontologies

There is not less wit nor less invention in applying rightly a thought one finds in a book, than in being the first author of that thought.

Pierre Bayle

3.1. Introduction

In chapter 2 we fixed the necessary terminology that will be used throughout this dissertation and briefly described the state of the art in ontology change and other fields relevant to this work. The rest of this thesis focuses solely on the problem of ontology evolution. We will assume that heterogeneity is not an issue, or that it has been somehow resolved using an ontology alignment or related algorithm. We will also ignore the issue of propagating the changes actively or passively (via an ontology versioning algorithm). This does not mean that we consider these problems less important or easier to address; however, they are irrelevant to our main research focus, which is the problem of identifying the changes that should be performed to an ontology when new information becomes available.

The purpose of this chapter is very different from that of chapter 2; in chapter 2, we simply described the current research status in ontology evolution and related areas (ontology change) without presenting the major problems in each field or any solutions to such problems. In this chapter we focus on ontology evolution and perform a more critical review on the proposed ontology evolution methods, uncovering certain weaknesses of the currently used research model which are likely to cause problems in the long run. Based on this analysis, we propose, justify and evaluate a certain approach based on the related field of belief change that will hopefully resolve these deficiencies. Finally, we describe some independently developed approaches ([KL04], [LM04], [MLB05]) which have recently appeared in the literature and closely follow our line of thought. As already mentioned, this part of our work will be kept at a fairly abstract level; our aim at this point is only to provide some general guidelines towards a research direction that will ultimately give us the desired solutions.

3.2. Weaknesses of Existing Ontology Evolution Tools

The Current Situation

Research on ontology evolution is still in its early stages [NM99a]; the field is a very active, but immature [NK04], research area of great importance for the Semantic Web. The current

state of the art in ontology evolution, as well as a list of existing tools that help the process can be found in [HS04]. Some of these tools are simple ontology editors, while others provide more specialized features to the user. In some cases, the user can define some kind of pre-defined "evolution strategies" [SMMS02] that control how changes will be made, thus allowing the tool to perform some of the required changes automatically. Other tools allow collaborative edits, i.e., several users can work simultaneously on the same ontology [DSWKB00], while others support transactional changes [HS04]. In other tools, features related to ontology versioning, undo/redo operations and other helpful utilities are supported [DSWKB00]. Some of these tools provide intuitive graphical interfaces that help the visualization of the process [LSV05].

Despite these nice features, there is a certain part of ontology evolution that is inadequately handled by current tools. The contribution of our work lies mainly on this "problematic" part, which resides in the second and third phase of ontology evolution, namely the change representation and the semantics of change phase [SMMS02]. Before these phases, the need for a change is identified; during these phases, we represent the change in a suitable format and determine what modifications must be made to the ontology in response to this need, as well as the indirect effects of these modifications on the validity, consistency and quality of the information stored in the ontology; if problems are detected with our initial modifications (e.g., if the removal of a class renders a certain instance unclassified), further changes may be required to restore the ontology to an "acceptable" state. Once all these changes have been determined, they can be applied to the ontology, propagated to dependent elements and validated (during the fourth, fifth and sixth phase respectively).

Unfortunately, current state-of-the-art approaches simply avoid the difficulty of dealing with these two phases (second and third) by letting the user handle them. To the author's knowledge, there is no tool acting like a "black box", i.e., a tool that receives the new fact and the ontology and returns the updated ontology in the output, without user participation. At best, there is some kind of pre-defined "evolution strategy" that controls how changes will be made or a declarative specification of the required changes using a suitable language [SMSS03]. In short, current work on ontology evolution resorts to ontology editors or other, more specialized tools whose aim is to *help* users perform the change(s) manually rather than to *perform* the change(s) automatically [HS04].

We believe that this approach will prove itself insufficient in the long run. Even though such tools and the features they include (change propagation, transactional properties of changes, undo/redo operations, collaborative edits etc) are undoubtedly useful, they need to be coupled with methods that allow the system to handle the whole cycle of ontology evolution by itself, without human intervention.

Of course, we are not claiming that human intervention should be avoided in every case; on the contrary, human participation in the process of ontology evolution should be encouraged and supported whenever possible in order to guarantee the quality of the resulting (evolved) ontology. On the other hand, there are several applications where human intervention is difficult, impossible, or simply undesirable; as a result, having the option to perform the changes in an automatic way is clearly a desirable feature of an ontology evolution algorithm.

In this respect, we believe that current research is misguided. In the rest of this section we will provide reasons for this claim and propose a direction for future research, based on the research area of belief change, that will, hopefully, lead to a solution to this problem.

First Problem: Manual Ontology Evolution

One problem with current approaches is that they require a varying level of human intervention in order to work properly. Some researchers consider this a necessary feature of the ontology evolution process [HS05a], [HS04], considering unrealistic the expectation that changes upon ontologies could be handled automatically [NM99a]. This conclusion is based on the argument that any given change can be resolved in several ways; moreover, the user requirements may be differing, even for the same change [SMSS03]. Thus, it is difficult for a computer system to decide on the best way to resolve a given change. Despite the validity of this claim, we believe that it is unrealistic to always rely on human participation during ontology evolution [SMMS02], for several reasons.

First of all, the human user that intervenes in the process should be an ontology engineer and have certain knowledge on the domain. Very few people can be both domain and ontology experts. But even for these specialized experts, it is very hard to perform ontology evolution manually [HS05a], [SMMS02]. So, in domains where changes occur often, it is simply not practical to rely on humans; the same holds in applications where it is difficult, impossible or undesirable for an ontology engineer to handle the change himself (as in autonomous robots, software agents or time-critical applications).

A manual evolution approach will, in many cases, perform better, in the sense of rationality, than any computer system devisable. However, there are exceptions to this rule. Different ontology engineers may have different views on how a certain change should be implemented [SMMS02]. These views may be affected by commonsense or background knowledge, personal preferences or ideas, subjective opinions on the domain and so on. This means that there is no single objectively correct way of updating an ontology in response to a certain change, but several subjectively correct ones. On the other hand, a computer-based evolution could, at least, guarantee determinism, objectivity and reproducibility of the results, even though some people may disagree on how a change was implemented. But then, is there a consensus on the effects of a given change even among humans? Do we revise our beliefs in the same way? If two ontology engineers are presented with the same ontology and the same change (or sequence of changes), is it certain that the final result (updated ontology) will be the same? We doubt that even humans can agree on the best way to perform a given change.

Another source of problems for manual ontology evolution is the complexity of modern day ontologies: such complex ontologies are usually developed by several ontology engineers or even teams comprising different expertise. A change in one part of the ontology might have unintended effects in other parts of the ontology [SMSS03]. The person who made the change may be unaware of the full content of his change's effects, as he doesn't know all the parts of the ontology. In fact, no single ontology engineer may be in position to determine the full effects of a change, unlike a well-designed computer system.

These points uncover the need for automatic ontology evolution; computer-based ontology evolution is not only *necessary* for many applications, it is also *desirable* in certain contexts. A human supervision by specialized experts should be highly welcome and should be encouraged whenever possible; however, the system should be able to work even without it.

Human intervention should constitute an optional feature guaranteeing the quality and effectiveness of the ontology evolution process; but it should not be a necessary one.

More specifically, an ontology evolution system should support the participation of the ontology engineer by means of always recording the changes it performed in full detail, along with justifications on the various choices. These should be available for review by the ontology engineer, allowing him to periodically check the quality of the resulting ontology. Whenever necessary, the ontology engineer should have the option to undo some, or all, the changes performed and/or perform changes of his own. In short, the role of the ontology engineer is not to perform the changes himself but to verify the quality of the system's output and possibly parameterize it, if he is not satisfied with the results. The most sophisticated current ontology evolution systems provide all these supporting features [HS04], but cannot perform the changes automatically. Therefore, the main gap that remains to be resolved is the automation of the process; this problem will be one of the main concerns of this thesis.

Second Problem: Too Many Operators

Another problem with the current research is related to the representation of changes (second phase of ontology evolution). In tools that are simple ontology editors, there is usually little or no support for any kind of complex changes to the ontology; the user can simply delete and add ontological elements, or, at best, move or copy them [HS04]. In more specialized tools for ontology evolution, there is a pre-defined set of elementary and/or composite operations that are supported, providing a greater flexibility to the user. For each such operation, there is an associated procedure that handles the change as well as the effects of the change (semantics of change phase); this procedure can, in some cases, be parameterized to cover different needs. Unfortunately, there is no guarantee that the provided parameterization is enough to cover any possible need of the knowledge engineer. Unforeseeable needs may require unforeseeable reactions to a given change.

Furthermore, as explained before, there is no limit on the number of complex operations that can be considered; even if we restrict ourselves to the most common (or interesting) types, there is usually a large number of them (in [SK03], for example, the authors report that 120 such operations have been defined and that the list is growing). This results in too many different operations (complex changes) to consider; creating a new procedure for each such operation is a non-scalable approach which increases the complexity of current ontology evolution tools, given the large number of useful operations.

The problem becomes even more complicated due to the fact that not all different types of change are readily available at design-time. New needs may require new operations. To handle operations that are not in the supported list, the ontology engineer should choose a sequence of two (or more) simpler (more elementary) operations to perform. Unfortunately, such a choice might affect the quality of the change and could lead to unforeseeable problems [SMMS02]. For example, deleting and then re-adding a concept in a different position in the hierarchy is not the same as moving the concept; in the former case, the instances of the concept will have to be removed or re-instantiated to a different concept; in the latter, no changes to the individuals will be required. In this simple example, the effects of the splitting of the operation in two more elementary operations may be easy to determine and/or undo; in more complex cases, this is far from trivial.

In any circumstance, this process cannot be performed without human intervention; it is impossible for a computer system to handle automatically a change that is not in its list of

supported operations, even if the unknown operation can be perfectly broken down in more elementary, supported operations. How could the system handle an operation it does not know? How could it select the operations to use for substituting the unknown operation? How could it evaluate the effects of its choice on the final result? If the system could handle the unknown operation or if it knew its semantics, then this operation would not be unknown in the first place! It could be handled normally and no substitution would be necessary at all. Such a problem is very likely to occur, as the creation of a complete list of composite operations is not possible [KN03]. Therefore, the current situation as far as the representation of changes is concerned practically overrules the possibility for automatic ontology evolution, which is a very important drawback for current tools, as explained in the previous subsection.

Third Problem: Lack of Adequate Formalizations

In current approaches, a change request is an explicit statement of the modifications to be performed upon the ontology; however, this request must be determined by the knowledge engineer in response to a more abstract need (e.g., an observation or a new fact). Thus, current systems do not determine the actual modifications to be made upon the ontology when faced with a new fact; they simply help the user determine them and feed them to the system for implementation.

Those systems that perform some of the tasks of evolution themselves (by suggesting the best way to resolve a certain change, or by implementing some of the simpler changes themselves) have been developed using certain heuristics based on the experience and expertise of the researchers (their creators) and ontology engineers; in effect, they attempt to emulate human behavior [SM02]. They are not theoretically founded and their theoretical properties remain unspecified. Moreover, no formal methods of evaluation have been determined; as a result, there is no formal characterization of which result is "best" given an ontology and a change, so we can't compare two ontology evolution algorithms in terms of effectiveness or rationality.

Given the current state of the research, such issues have been neglected, because the decisions on the changes to be made are left upon humans. This way, empirical decisions based on the expertise of the person who performs the change are taken at the time when evolution is performed: whenever the ontology engineer is faced with a change, he decides on his alternatives and selects the "best" one, which is then fed to the system for implementation. All these decisions are based on his expertise on the subject, not on a formal, step-by-step, exhaustive method of evaluation.

However, to develop fully automatic ontology evolution algorithms, several questions need to have a definite, formal answer. For example, how can one track down all the alternative ways to address a given change, using a formal and exhaustive process? How can one guarantee that there are no further alternatives for a given change, except the ones found? How can a computer system decide on the "best" of the different alternatives? Most importantly, what is the definition of "best" in this context? What are the features or properties that make a certain result of a change "better" than another? Are there any properties that should be satisfied by a "good" ontology evolution algorithm?

Unfortunately, resolving these issues in a general manner is not easy using the current research direction because each type of change is treated differently, using a stand-alone, specialized and informal process. Unless a more formal path is taken, the ontology evolution research is doomed to never find answers to these questions, and, consequently, to never

escape the need for an ontology engineer supervising and participating in the process of ontology evolution. The opposite is true as well: the participation of the ontology engineer in the process of ontology evolution jeopardizes any formalization attempt, because it introduces non-determinism and non-reproducibility of the results.

3.3. Our Proposal

General Guidelines Towards a Solution

Our analysis uncovers the need for the development of an automatic, unifying and formal method for describing and handling the process of ontology evolution. A carefully designed formalism will allow us to represent this process in an adequate manner; it will also allow the system to evaluate the situation itself and choose automatically how the required change should be made.

Thus, the basic step towards resolving the aforementioned deficiencies is to address the problem of ontology evolution in a more formal manner. This will allow a formal study of the intuitive processes behind the dynamics of change, which, in turn, will lead to a deep understanding of the various processes involved and eventually allow the development of a formal, unifying and automatic approach to the problem. This implies that all three deficiencies can be actually resolved through the development of an adequate formalism, so all three problems are actually interlinked.

More specifically, we believe that the task of resolving the outlined weaknesses of the currently used research model passes through two major phases:

1) The development of a formal, unifying method of representing and handling the ontology, the change and the process of ontology evolution; this presupposes the formal description of all the possible ontology evolution operations using a small, pre-defined set of processes (operations), as well as the development of adequate formalizations that will make the study of the problem as simple as possible, without depriving it from any of its peculiarities or properties.
2) A formal study on the processes behind ontology evolution, as well as on the dynamics of change in general; this will allow the deep understanding of these processes and ultimately allow us to loosen up the dependence of the ontology evolution process on the knowledge engineer. This formal study cannot be made without the formalizations already developed during the first phase.

The Proposed Solution

The focus of this chapter lies mainly on the first phase of the above process. More specifically, we will present a logic-based formalization with the desired properties which is heavily influenced by the related field of belief change [Gar92a]. Our purpose at this stage is to develop the formalizations necessary to allow the formal study of the ontology evolution field, not to propose concrete solutions to the problem.

This does not mean that the second part of this two-phase process is neglected. One of the major advantages of our formalization is that it allows us to link the problem of ontology evolution with that of belief change. Belief change deals with the automatic adaptation of a KB to new knowledge [Gar92a]. Therefore, the focus of belief change is on determining the most rational ways of dealing with changes on knowledge, as well as on the development of algorithms that update KBs automatically. This is exactly what the current ontology evolution

research is missing: the development of formal results related to the "proper" way to perform the required changes and algorithms implementing such results in an automatic, effective and efficient way. Therefore, our proposed formalization will allow us to exploit previous work on belief change in order to formally study the dynamics of change and ontology evolution (i.e., the second phase of the above process), ultimately leading to the development of a formal, unifying and automatic approach to the problem.

Unfortunately, works related to belief change are not directly applicable to ontologies, as they are based on knowledge representation languages which are not used in ontologies. Thus, our approach consists in studying both fields in an attempt to identify common grounds and connections between them; this will allow the migration of certain techniques used in belief change into ontology evolution. Moreover, we will revisit the main concepts and ideas dealt with in belief change under the prism of ontology evolution and use the ideas expressed in belief change as an inspiration for expressing similar ideas in the context of ontologies and the Semantic Web.

Formalization: Basic Ideas

In order to develop an adequate formalization for the problem of ontology evolution, we will have to use a different view of an ontology based on its logical properties. A formal semantics and terminology for ontologies will be required; this will be provided by DLs [BCMNPS02], which will arguably play a very important role in the development of ontologies and the Semantic Web [BHS03]. The main advantage of DLs in this respect is that they provide a well-founded formal semantics, as well as enough diversity to cover very different needs; these desirable properties have triggered a significant amount of research on the subject, allowing a good understanding on the properties of this family of languages. For these reasons, in the rest of this dissertation, we will assume that ontological knowledge is expressed using some logic from the DL family; despite this assumption, many of our general ideas are transferable to other formalisms for ontology representation as well.

Representation of Ontologies

Obviously, the most fundamental decision involved is how the ontologies themselves should be represented. Most ontology editors (such as those presented in [MFRW00], [NM00], [LSV05]) use a graphical tree-like interface to represent and manipulate ontologies [HS04]. This allows the user to actually view the changes performed. This graph-based viewpoint regarding the ontology representation is pervasive in most ontology evolution algorithms, as it affects the decisions on how each change should be implemented. Alternative graphical representations have also been proposed [TXZZZY05].

Graphical representations are extremely useful for visualizing the way that the domain conceptualization was implemented in an ontology. They also help novice users and domain experts get acquainted with the field and understand the conceptualization, by hiding much of the semantic and syntactic complexity of the ontology behind intuitive interfaces and simple visual metaphors [NKKM04]. However, they are often not expressive enough for complex applications, because certain complex DL axioms cannot be easily expressed using a graph; for example, one cannot easily express in a graph the axiom:

$$A \sqcap B \sqsubseteq \forall R.\exists S.(A \sqcup \neg D) \sqcap C$$

Similarly, axioms that use complex connectives, such as $\sqsubset$, $\not\sqsubseteq$ etc cannot be directly represented. This might be a problem when one wants to express this kind of information.

Current ontology evolution approaches are based on the structure of the ontology in a graph-based representation, which has shifted the focus of the relevant research to concepts, roles, individuals and how they are structured in the ontology graph; in effect, most existing work on ontology evolution builds on frame-like or object models [HS05a]. Arbitrary axioms are often not considered part of an ontology and little or no attention is paid to them in ontology change [NM99a]. This results in the loss of much of the expressiveness of the underlying language, disallowing, for example, the definition of constraints like the above.

For knowledge engineers and ontology experts, an algebraic representation provides a more concise and formal representation of the conceptualization, has a cleaner semantics and allows easier formal manipulation than the graph-based approach. In fact, a combination of the approaches usually works best, as it allows us to use the best of both worlds.

Under the algebraic approach, the knowledge of the ontology is represented as a pair <S,A>, where S is the *vocabulary* (or *signature*) containing information on the elements appearing in the ontology (concepts, roles, individuals) and A is a *set of ontological axioms* [KS03]. The vocabulary may be a single unstructured set containing all the concepts, roles and individuals relevant to the ontology, or it may have some structure denoting, for example, the concept hierarchy. The set of ontological axioms may contain an arbitrary number of axioms; in our case, these should be axioms of the DL used to represent the ontology.

In this work, we will use a simplification of the algebraic approach, by dropping the signature structure and *representing an ontology as a set of DL axioms, under a given, predefined DL* (DL KB). This way, our approach focuses on axioms, following the axiom-centered ontology model [HS05a], ignoring the signature of the ontology. In the following, the terms ontology and DL KB will be used interchangeably; furthermore, we will not generally force any particular properties on the underlying DL, unless otherwise specified.

Since the graphical structure of the ontology can be completely determined by a set of axioms, we believe that viewing the ontology as a set of axioms provides a more general method of ontological representation, allowing the easy switch to the graphical representation (with some possible loss of expressiveness) if necessary. Furthermore, the logical representation is simpler and more straightforward than the classical algebraic approach, and, as we will see, it allows us to solve one of the main problems in current ontology evolution research that we described before, namely the lack of a unifying method of treating the different kinds of changes.

Most importantly, this viewpoint of DLs (and ontologies) allows us to introduce a common formalism in terms of which both classical logic and DLs can be described, namely Tarski's logical model, which was described in section 2.5. This fact facilitates the task of migrating belief change techniques (mostly based on classical logic, which is also describable using Tarski's model) to the DL framework.

It can be easily shown that all monotonic DLs (which are the focus of this thesis) can be engulfed by Tarski's framework, by setting the set L to contain exactly the axioms that said DL allows for and $\vdash$ to be the inference relation of the underlying DL under its standard model-theoretic semantics [BCMNPS02]. Under this formalization, a DL is represented by a pair <L,Cn> (equivalently: <L,$\vdash$>) and an ontology (i.e., DL KB) is represented by a single set O$\subseteq$L, where L is the set of expressions of the underlying knowledge representation DL <L,Cn>. This is quite similar to the belief change approach, where a KB is represented by a set K$\subseteq$L, defined over a representation language <L,Cn>.

The main disadvantage of our proposed ontological model is that we lose the information normally stored in the signature of the ontology. This is not as major a problem as it seems, because most of the information in S can be represented using axioms as well. For example, if S is a set equipped with a partial ordering representing a certain hierarchy between concepts, then the hierarchy information can be expressed in the form of axioms in the set O using the subconcept connective of DLs ($\sqsubseteq$). Things become more complicated when one tries to denote the "relevance" of certain elements (classes, roles or individuals) to the ontology. This is normally denoted by including the relevant element to the signature of the ontology, but cannot be expressed using a set of DL axioms. This problem will be resolved at a later point in this section, using the concept of the *Closed Vocabulary Assumption* (*CVA*) and the *Existence Assertion Operator*.

Here is a simple example showing how a simple ontology can be represented in graphical, algebraic and logical format:

Example 3.1. Ontology: BIRD_ONTOLOGY

BIRD$\sqsubseteq$FEATHERS	(all birds have feathers)
BIRD$\sqsubseteq$FLY	(all birds fly)
PENGUIN$\sqsubseteq$BIRD	(all penguins are birds)
PENGUIN$\sqsubseteq\forall$EAT.FISH	(penguins eat only fish)
BIRD(TWEETY)	(TWEETY is a bird)

The logical representation of the ontology is a set containing the above axioms. This information can also be represented in a graph as follows:

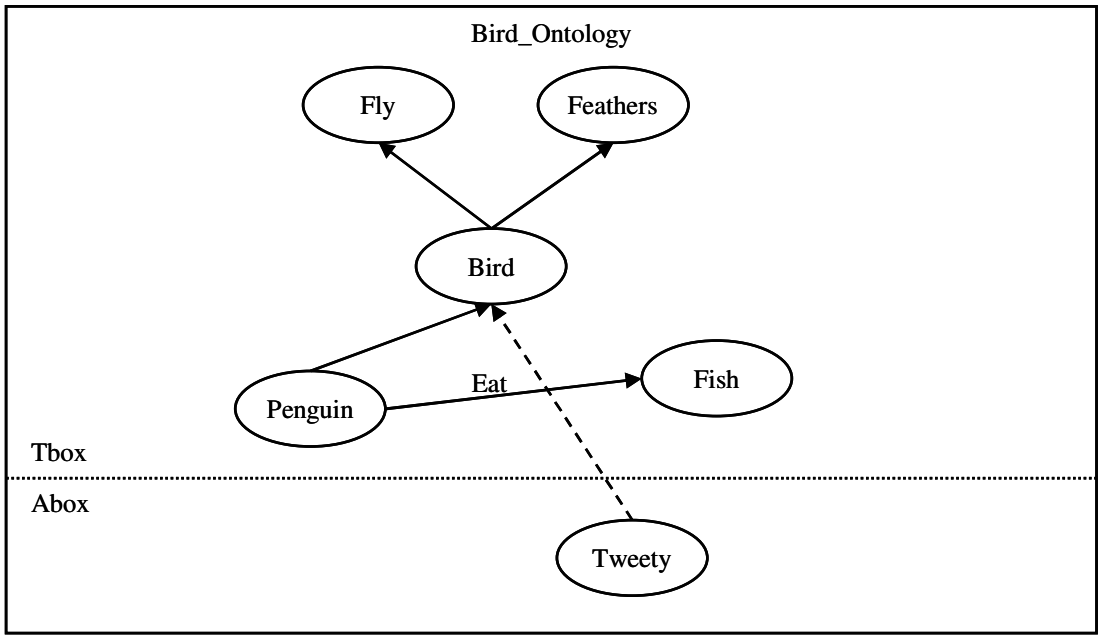

Figure 3.1. The visualization of BIRD_ONTOLOGY

The algebraic representation is composed of two structures: the signature S and the axioms A. The set of axioms A contains all the above axioms (as in the logical representation); the set S contains all the elements that are relevant to the ontology; one such S could be:

S={BIRD, FEATHERS, FLY, PENGUIN, FISH, EAT, TWEETY}

Change Representation Phase

As already mentioned, one of the major drawbacks for current ontology evolution research is the great number of different complex operations that can be defined [SK03], each with its own handling process. This fact jeopardizes the development of proper formalizations for the process of ontology evolution and hinders our effort to define automatic ontology evolution processes. Thus, we are forced to drop the current viewpoint regarding the representation of the changes; we need a method that will allow us to represent all possible changes in a uniform way. This need has already been identified in the literature, as the desire for a new declarative language for the specification of a change request [HS04]; an example of such a language, for RDF, has been proposed in [MSCK05].

We argue that this language need not be different from current ontological languages. In fact, we propose the use of the same abstraction that we proposed for ontology representation, i.e., to *represent the change as a set of DL axioms*. The DL representing the change should be the same as the one used for the representation of the ontology. This way, if <L,Cn> is the logic used for ontology representation, a change can be any set $X \subseteq L$.

This requirement does not clarify the semantics of this set. Usually, this set will contain certain axioms which should be implied by the ontology after the change. But this is not always true, as there are cases when one needs to remove knowledge from the ontology; in this case, the information provided (i.e., the set of DL axioms) is a set of facts that *should not be implied* by the resulting ontology. Furthermore, notice that ontology evolution refers to the process of modifying an ontology in response to a certain change in the domain *or* its conceptualization; our naïve approach ignores the difference between these two types of changes. This difference has already been studied in the belief change literature: it is the same as the fundamental philosophical difference between a change that refers to a change in the representation of the domain (i.e., revision and contraction in the belief change terminology) and a change that refers to a dynamic change in the domain (i.e., update and erasure in the belief change terminology).

To cover these diverse needs, we propose, inspired by the belief change literature, the introduction of four different ontology evolution operations, namely *ontology revision*, *ontology contraction*, *ontology update* and *ontology erasure*. We could also introduce ontology expansion, but the operation of expansion is rather trivial, so we chose to ignore it to keep the model simple. In all these cases, the change is represented using a set of DL axioms; however, this set of DL axioms has different semantics in each operation. This causes the splitting of the ontology evolution research in four different areas, each dealing with a different type of change (operation). This is similar to the belief change field, where belief change is split in four different, but closely related, subfields, namely belief revision, contraction, update and erasure. It is also a great example of how the intuitions expressed in belief change can be applied to ontology evolution, saving us from potential pitfalls.

More specifically, we *define ontology revision as the process of incorporating in an ontology some piece of knowledge regarding a static world*. Similarly, *we define ontology contraction as the process of retracting from an ontology some piece of knowledge regarding a static world*. Analogously, *ontology update is defined as the process of incorporating in an ontology some piece of knowledge regarding a dynamic world*, while *ontology erasure is defined as the process of retracting from an ontology some piece of knowledge regarding a dynamic world*.

Operation	Type of Change (Addition/Retraction)	State of the World (Static/Dynamic)
Ontology Revision	Addition	Static
Ontology Contraction	Retraction	Static
Ontology Update	Addition	Dynamic
Ontology Erasure	Retraction	Dynamic

Table 3.1. Ontology evolution operations

Table 3.1 shows the properties of each of the four operations. When the type of change is an addition (revision, update), we have the incorporation of new knowledge in the ontology; in effect, the set of DL axioms of the change should be implied by the resulting ontology. When the type of change is a retraction (contraction, erasure), the set of DL axioms should not be a part of the resulting ontology, neither explicitly, nor implicitly (i.e., deducible by the ontology). When the state of the world is static (revision, contraction), we have some new observation, experiment etc regarding a world that has not changed; in effect, our knowledge is enhanced with new data, allowing us to improve the conceptualization of the world. When the world is dynamic (update, erasure), there is nothing wrong with our conceptualization, but the world itself is evolving; therefore, our conceptualization of the world should adapt to the new reality.

Representing changes at the level of axioms provides a very fine-grained approach to the problem [HS05a]. It also saves us from the need to define complex operators for each type of change imaginable, since all the changes can be now expressed using one of the four operations and a set of DL axioms; if the change (i.e., the captured fact to be included in the ontology) cannot be expressed using a set of DL axioms, then the change is beyond the expressive power of the underlying DL, so we can't express it in the ontology anyway. In such a case, the problem lies on the representation language chosen, not on our approach.

In the ontology BIRD_ONTOLOGY (example 3.1), if we lose our faith in the belief that all birds can fly, then we would like to remove the IsA relationship $\text{BIRD} \sqsubseteq \text{FLY}$. In this case, we should contract with the set $\{\text{BIRD} \sqsubseteq \text{FLY}\}$. If, on the other hand, we observe that penguins actually don't fly and that TWEETY is not a penguin then we would like to revise with the set $\{\text{PENGUIN} \sqsubseteq \neg\text{FLY}, \neg\text{PENGUIN}(\text{TWEETY})\}$. Now, let us present a more concrete example that will allow us to show the differences between all four types of ontology evolution:

Example 3.2. Consider a box containing several objects, some of which are chess pieces; all chess pieces are either black or white, but all black pieces are made of plastic and all white pieces are made of wood. We also know that there is at least one chess piece in the box, call it KING. These facts are represented in an ontology as follows:

Ontology: CHESS_PIECES

$\text{CHESS_PIECE} \sqsubseteq \text{BLACK} \sqcup \text{WHITE}$	(all chess pieces are black or white)
$\text{BLACK} \sqcap \text{CHESS_PIECE} \sqsubseteq \text{PLASTIC}$	(all black pieces are plastic)
$\text{WHITE} \sqcap \text{CHESS_PIECE} \sqsubseteq \text{WOODEN}$	(all white pieces are wooden)
disj(BLACK,WHITE)	(nothing can be both black and white)
disj(PLASTIC,WOODEN)	(nothing can be both plastic and wooden)
CHESS_PIECE(KING)	(KING is a chess piece)

Some examples of changes are:

{CHESS_PIECE(QUEEN)}	(QUEEN is a chess piece)
$\{\text{BLACK} \sqsubseteq \text{CHESS_PIECES}\}$	(all black objects in the box are chess pieces)

{PLASTIC(KING), PLASTIC(QUEEN)}	(both KING and QUEEN are plastic)
{BLACK(KING)}	(KING is black)

Each of these changes may have different semantics depending on the operation involved; in other words, each of these changes has four different "intended results" depending on the ontology evolution operator chosen. Let us review these different cases for the latter change (KING is black), by describing one possible situation under which each operation is applicable as well as the effects that this change should have upon the ontology in each case:

- *Revision*: suppose that, through observation, we learn that KING is black. The proper change in this case should be a revision with the set {BLACK(KING)}. This would normally require no further changes to the ontology, except from the addition of the new fact: BLACK(KING); thus, the new ontology is the union of the original (CHESS_PIECES) with the change ({BLACK(KING)}). In effect, the new fact allowed us to refine our knowledge regarding the chess piece KING. Let us name this new ontology CHESS_PIECES_NEW.
- *Contraction*: suppose that, starting from the revised ontology (CHESS_PIECES_NEW), we lose our faith in the belief that KING is black, because, for example, the observation was made by an unreliable person who could be lying. This is a contraction operation, which, once again, should be represented using the expression {BLACK(KING)}. This contraction can be easily accommodated by simply removing the above statement from the ontology. If, on the other hand, this contraction was performed upon the original ontology (CHESS_PIECES), then it would have no effect, as the axiom BLACK(KING) is not a consequence of the ontology CHESS_PIECES in the first place.
- *Update*: suppose now that we have the original ontology (CHESS_PIECES) and paint KING black; then the change is an update (because the domain itself changed). This update should be represented using the same set: {BLACK(KING)}. But the result of the change is very different this time: notice that, if the piece was originally white, then it would be wooden. But now it is painted black; thus, there could exist one chess piece that is both black and wooden. This analysis shows that the rule BLACK $\sqcap$ CHESS_PIECE $\sqsubseteq$ PLASTIC should no longer be valid after this update.
- *Erasure*: suppose that we send a robot in the box with the following order: "if KING is black, paint it with an arbitrary color; otherwise, do nothing". Thus, if the color of KING is not black, then no action will be taken. If its color is black, then the robot may choose to paint it any color, including red, green, or even repaint it black. Thus, all we can assume by the robot's action is that the belief that the color of KING is black should be removed; this is an erasure operation (because the world is changing dynamically), represented by the set {BLACK(KING)}. We can't assume that KING has (or does not have) any particular color, as we don't know the color that the robot selected to paint it, so update is not applicable here. This action may invalidate some of the statements of our ontology; indeed, we may now have a red chess piece, or a chess piece that is both white and plastic. Thus, the axioms CHESS_PIECE $\sqsubseteq$ BLACK $\sqcup$ WHITE and WHITE $\sqcap$ CHESS_PIECE $\sqsubseteq$ WOODEN are both invalidated.

If we had fed example 3.2 to one of the current ontology evolution tools, we would have to take the literal description of the change (for example: "KING was painted black") and then decide on the complex and/or elementary changes required. In the particular case in which "KING was painted black" (i.e., the third bullet of our example), two changes would be required. First, we should remove one IsA:

$$\text{BLACK} \sqcap \text{CHESS_PIECE} \sqsubseteq \text{PLASTIC}$$

Then, we should add the assertion statement:

$$\text{BLACK}(\text{KING})$$

Alternatively, we could have chosen to weaken, instead of removing, the axiom:

$$\text{BLACK} \sqcap \text{CHESS_PIECE} \sqsubseteq \text{PLASTIC}$$

This could be made, for example, by replacing it with the following IsA:

$$\text{BLACK} \sqcap \text{CHESS_PIECE} \sqcap \neg\{\text{KING}\} \sqsubseteq \text{PLASTIC}$$

The latter statement asserts that if something is a black chess piece, but not the particular chess piece KING, then it is plastic. Such a statement would be more accurate in this situation. And there are other possibilities too.

Therefore, even in this simple example, there is a substantial amount of effort required by the ontology engineer in order to accommodate the change. The ontology engineer might have failed to notice the IsA removal required; such a failure would not lead to any inconsistencies, so the system could not have noticed the problem and it could take a long time before the ontology engineer noticed the problem himself. Even if he had noticed the problem in the first place, he could have chosen to remove the IsA altogether, failing to notice the second, more accurate alternative of weakening it.

In the alternative approach that we propose for the representation of the changes, the ontology engineer would need to decide on the operation required and to represent the change using a set of DL axioms (in this particular case: "update with {BLACK(KING)}"). So, he would only need to feed the system with the input ontology and order it to update this ontology with the expression {BLACK(KING)}. The system should then (somehow) perform the update in a proper and automatic way. The problem of the system deciding on the changes to be performed is very difficult, but there are several belief change techniques that could be of use here; we will deal with this part of the problem in later sections. The important thing with these techniques is that they act in a formal and deterministic manner, eliminating the possibility of mistakes like the ones described above.

In effect, what we are proposing is an extra layer in the system that will perform the actions currently handled by humans (phases two and three of ontology evolution). Once the problem of deciding the modifications to be made is solved, the system could then (automatically) forward the results to the next phase for implementation; from this point on, the process can be automatically handled by existing systems. Therefore, our approach is not intended to replace existing tools, but to supplement them; we hope that current tools will use this approach to improve their effectiveness by being able to perform the changes in an automatic manner.

Of course, even in our approach, there is some user interaction that is required; however, this could be easily avoided if necessary. For example, a robot can reasonably assume that all information it receives through its sensors are revisions; sensors should have inbuilt logic that allows them to translate observations (like the observation that a particular chess piece is black) to logical statements (i.e., {BLACK(KING)}). Once this is accomplished, the whole process can be fully automated. But even if this is not the case, the human interaction required is substantially less than the one required in standard ontology evolution approaches and does not require familiarity with the underlying ontology.

Change Representation Phase: Facts Versus Modifications

It is easy to notice that there is no exact mapping between the four operations we propose for ontology evolution and the operations that are being used in the ontology evolution literature. The reason for this lack of mapping between these two sets of operations lies on a fundamental difference underlying their definition: the two approaches reflect a different viewpoint on how a change should be interpreted and handled, which renders them incomparable.

Our approach is "fact-centered": a new fact represents a certain need for the evolution of an ontology. The ontology engineer (or some automatic sensor or similar device) should identify the type of the new fact, i.e., whether it changed the real world or not and whether it added knowledge or added uncertainty by casting doubt on some existing knowledge (in which case we are dealing with removal of knowledge). Addressing these two issues is enough to determine the type of operation involved; the exact operand of the change (i.e., the set of axioms representing the change) should also be determined, manually or automatically. These two facts constitute the change in our framework; this change is then fed into the system which should identify the actual modifications to perform upon the ontology to address the new fact and perform these modifications automatically.

The standard approach is "modification-centered": we are not interested in the fact itself that initiated the change; rather, we are interested on the actual modifications that should be physically performed upon the ontology in response to this new fact. In this case, the ontology engineer identifies the new fact and decides (manually or with the aid of the system) on the modifications that should be performed upon the ontology; these are ultimately fed to the system for implementation. This gives more control to the ontology engineer, making the process more straightforward, but it also adds a significant overhead to the problem of ontology evolution, making it more time-consuming and error-prone in several cases [SMSS03].

This analysis shows that the two approaches are not directly comparable, as they are based on a different paradigm; as a result, the comparison of the results of a standard ontology evolution tool (modification-centered approach) with the results of a tool based on some belief change technique (fact-centered approach) would not make much sense.

The fact-centered approach realizes the extra layer of abstraction discussed in the last subsection: the modifications to be performed upon the ontology are decided by the system, not by the ontology engineer. This allows the ontology engineer to deal with high-level facts only, leaving the low-level modifications that should be performed upon the ontology in response to these facts to be determined and handled by the system. This forms another argument in favor of the non-comparability of the two approaches, as they act on different levels of abstraction. It is also important to notice that, in the fact-centered paradigm, the change can be determined without knowing the contents of the ontology, a property which simplifies this process.

Can I Remove the Penguin?

Unfortunately, there is a certain deficiency in our method. One common type of operation that occurs in ontologies is the addition or deletion of concepts, roles and individuals. In the standard algebraic approach, this amounts to changing the vocabulary of the ontology. However, the vocabulary has been dropped in our logical framework, so such kinds of changes cannot be directly expressed.

This problem is actually a different facet of the problem that was outlined in a previous subsection regarding our inability to denote the relevance of certain elements to the ontology using axioms of standard DLs. Indeed, both problems actually stem from the fact that DLs do not contain any constructors allowing us to express facts like "concept PENGUIN exists in the ontology". Because of this, no set of DL axioms could express a change like "remove concept PENGUIN" or "add concept PENGUIN". Since such facts cannot be expressed in the ontology either, this is actually a problem of the representation language. We will propose two methods to deal with this deficiency. These methods use the concepts of *Closed Vocabulary Assumption* (*CVA*) and *Open Vocabulary Assumption* (*OVA*), as well as the *Existence Assertion Operator* (denoted by %).

Closed Vocabulary Assumption and the Existence Assertion Operator

The latter issue is based on the following fundamental (and usually implicit) assumption: that the ontology uses *exactly* the namespace elements which are relevant to the conceptualization of the domain. For example, a University ontology could refer to the namespace elements STUDENT, CLASS, COURSE, ENROLLS etc. On the other hand, the concept PENGUIN is probably not relevant to a University ontology. This assumption is very important for our purposes, so we will give it a special name: the *Closed Vocabulary Assumption* (hereby CVA).

In order for CVA to make sense, there should be a way to identify the elements which are relevant to the ontology from those which are not. This is normally made using the signature structure; however, under our formalism, this discrimination is not possible, as the signature has been dropped. One naïve method of dealing with this issue would be to assume that the relevant elements are exactly those that appear in at least one axiom of the ontology. This is a common assumption in the literature, where the signature file is often implicitly defined this way.

This method has two major deficiencies: first, it could be the case that some namespace element is relevant to our conceptualization, yet our knowledge is still incomplete and we have no clue on the properties of this element; in such a case, we would like to denote explicitly that this element is relevant to our ontology without using it in any axiom. Second, this method does not allow us to remove or add namespace elements from our ontology, so the problem of the operations that cannot be expressed in our framework remains.

The solution that we propose to resolve both these deficiencies is the introduction of a new operator that explicitly denotes the relevance of a namespace element to an ontology. This is the *Existence Assertion Operator* (%), which is a modal operator that can be used to enhance the expressive power of DLs by allowing the expression of such facts. In effect, the symbol % is a DL connective which is followed by one element from the DL namespace to form an axiom. The intuitive meaning of such an axiom is that said element "exists" in the ontology (i.e., it is relevant to our conceptualization of the world, as expressed by the ontology); for example, the axiom %PENGUIN explicitly states that the concept PENGUIN exists in the ontology and is useful for (i.e., relevant to) our conceptualization. This amounts to saying that PENGUIN is part of the ontology's vocabulary (signature). Notice that such an axiom gives no further information on PENGUIN; we cannot deduce any useful properties from it, except from the fact that PENGUIN is relevant to the ontology.

The addition of this modal operator is a direct improvement to the expressive power of a DL and is applicable to all DLs. It eliminates the need for a special signature structure in

ontologies, as we can now explicitly denote the elements that are relevant to the ontology (i.e., the elements that are in the signature of the ontology); it also resolves the problems with our initial naïve approach.

Notice that the fact that a certain namespace is relevant to our ontology can be now expressed either explicitly (using the operator %) or implicitly (by forming an axiom that contains the element). Formally, an element A "exists" in O iff $O \vdash \{\%A\}$, i.e., iff O implies its existence. This modal operator also allows the expression of the problematic operations: in order, for example, to remove an element A from an ontology O, one should perform a contraction operation (or erasure, depending on the context) with the set {%A} (i.e., $O - \{\%A\}$). An additional advantage of this approach is that it allows us to express all kinds of change operations, including those related to the "relevance" of elements, in a uniform way, using axioms only and without changing the proposed framework; this is in accordance to our general objective of keeping our framework simple and uniform.

In order to make this modal operator useful, we must equip it with formal semantics constraining how it can be used in the reasoning processes of the underlying DL. One initial idea towards the introduction of a formal semantics for % is the fact that the concept of "relevance" (or "existence") is directly related to the namespace elements that are "used" by the axioms of the ontology; for example, the axiom $A \sqsubseteq B$ should imply the "relevance" ("existence") of both A and B, because A and B are used in the axiom (so they are relevant to it, and, consequently, to an ontology that contains this axiom). Thus:

$$\{A \sqsubseteq B\} \vdash \{\%A, \%B\}.$$

To describe such types of implications formally, we will need an exact, formal definition of the elements that are "used" by an axiom, term or DL KB:

Definition 3.1. Let <L,Cn> be some DL and X be any expression of this DL (namespace element, term or axiom). The set of elements used by X is denoted by U(X) and is defined inductively as follows:

- If X is an element of the namespace ($X \in N_{DL}$), then $U(X)=\{X\}$
- If X is a term of the form $o(X_1, X_2, \ldots, X_n)$, where o is some n-ary operator of the DL ($o \in O_{DL}$, expressed in prefix notation), then $U(X)=U(X_1) \cup U(X_2) \cup \ldots \cup U(X_n)$
- If X is an axiom of the form $c(X_1, X_2, \ldots, X_n)$, where c is some n-ary connective of the DL ($c \in C_{DL}$, expressed in prefix notation), then $U(X)=U(X_1) \cup U(X_2) \cup \ldots \cup U(X_n)$

The definition can be extended to sets of axioms by setting $U(P)=\cup_{X \in P} U(X)$, for any set of axioms $P \subseteq L$.

Notice that, for an ontology O, the set U(O) corresponds to the signature of the ontology O (under the standard terminology). Obviously $U(X) \subseteq N_{DL}$. We will assume, by convention, that the union of a zero number of sets is the empty set (so $U(\varnothing)=\varnothing$ and $U(\top)=\varnothing$, for example). Using the concept of the elements that are "used" by an ontology, the semantics of the existence assertion operator can be defined relatively easily.

Two questions need to be answered in this direction. The first is, when is an axiom of the form %A implied? The answer to this question is intuitively simple: an axiom of the form %A is implied by a DL KB iff A is used by the DL KB. Of course, there might be elements which are relevant to the DL KB, yet they are not used by any axiom; if this is the case, such elements should be made explicit by adding existence assertion axioms forcing their relevance to the DL KB. Take for example the DL KB:

$$\{A \sqsubseteq B \sqcap C, D \sqsubseteq C\}$$

This DL KB should imply, among other things, the relevance of A, B, C and D, thus:

$$\{A \sqsubseteq B \sqcap C, D \sqsubseteq C\} \vdash \{\%A, \%B, \%C, \%D\}$$

However, it should not imply the existence of any other element (e.g., %E); such an implication would be contrary to the idea behind the introduction of CVA. If the ontology engineer thinks that E is relevant to the conceptualization of the ontology, he should denote this explicitly by adding the axiom %E to the ontology (notice that $U(\%E)=\{E\}$). In conclusion, the answer to our first question is that an axiom of the form %A is implied by a DL KB (say O) iff A is used by this DL KB (i.e., iff $A \in U(O)$).

The second question to be addressed is the exact opposite: what does an axiom of the form %A imply? The answer here is also relatively simple intuitively: nothing. The only thing that should be implied by such an axiom is the relevance of the involved element to the ontology. This "existence" does not give any further information on the element; its further properties are unknown. Thus, only trivial statements should be implied by existence assertion axioms.

Now consider a DL <L,Cn>. The two intuitive results that were presented informally above, can be formally expressed using the following two postulates:

(%1) If $P \subseteq L$ then: $P \vdash \{\%A\}$ iff $A \in U(P)$

(%2) If $N \subseteq N_{DL}$ and $\cup_{A \in N}\{\%A\} \vdash P$, then: $P \cong \cup_{X \in U(P)}\{\%X\}$

The first postulate constrains that an axiom of the form %A is implied by a set P iff A is used by P. The second postulate says that if a set is implied by a set of existence assertion axioms, then this set carries no interesting information, so the only thing it implies is the existence of all the elements it uses. This means that a set containing only existence assertion axioms is as close to a tautology as can be; all the sets that are implied by such sets are actually equivalent to some set which has only existence assertion axioms. Another, more intuitive way of viewing the meaning of the latter postulate will appear using the connection of the lattice theory with Tarski's framework that we will describe in section 4.5 (see figure 4.8 in page 97).

In the proposed version of the existence assertion operator, a statement of the form %A does not indicate whether A is a concept, role or individual, so one could consider refining the operator in order to allow the discrimination between elements that are concepts ($\%_C$), roles ($\%_R$) and individuals ($\%_I$), in effect replacing % with $\%_C$, $\%_R$, $\%_I$. Under this viewpoint, $\%_C A$ would mean that A "exists" (as part of the ontology's vocabulary) and is a concept, $\%_R A$ would mean that A "exists" and is a role and $\%_I A$ would mean that A "exists" and is an individual.

Unfortunately, it is quite hard to generalize the semantics of the single operator to capture the semantics of the more complex case with the three operators, because such a semantics would need to capture more facts. One should make sure that something is of one type only; for example, $\%_C A$ should not imply $\%_R A$ and vice-versa. Furthermore, one should be able to correctly determine the type of a certain namespace element in an axiom; for example the axiom $\forall R.A \equiv B$ implies that R is a role, while A and B are concepts, i.e., it implies that: $\%_R R$, $\%_C A$, $\%_C B$. To correctly determine this fact, one must take into account the semantics of the DL operator $\forall$ and connective $\equiv$. The same holds for all operators and connectives admitted by the DL at hand. Finally, the two conditions of the single operator (%) should be generalized to apply to all three operators ($\%_C$, $\%_R$, $\%_I$). Capturing all these facts would require more than a dozen postulates!

We believe that the introduction of these additional operators may be an overkill for most applications. As already mentioned in section 2.3, the namespace of a DL (N_{DL}) is assumed to be split into three disjoint sets (N_{DL}^C, N_{DL}^R, N_{DL}^I), which allow us to tell in an unambiguous manner whether a certain namespace element is a class, role or individual. This eliminates the need for three different modal operators and allows us to use the former approach, with the single operator, which is much simpler and has much cleaner semantics. For this reason, we will only use the simple approach in this thesis.

We will conclude this subsection with a simple lemma that uncovers certain properties of %; these properties can be shown using the postulates (%1), (%2) alone:

Lemma 3.1. Assume a DL <L,Cn>. Then the following hold:

1) For any $A \in N_{DL}$, it holds that: $U(\%A)=\{A\}$
2) For any $P \subseteq L$, it holds that $Cn(\cup_{A \in U(P)}\{\%A\}) \subseteq Cn(P)$
3) For any $P, Q \subseteq L$, $P \vdash Q$ implies $U(P) \supseteq U(Q)$
4) For any $P, Q \subseteq L$, $P \cong Q$ implies $U(P)=U(Q)$
5) For any $P \subseteq L$ it holds that $U(P)=U(Cn(P))$

Proof

The first fact is obvious by the definition of U.

For the second fact note that by (%1) for any $A \in U(P)$ it holds that $P \vdash \{\%A\}$, thus $P \vdash \cup_{A \in U(P)}\{\%A\}$. The last relation implies $Cn(\cup_{A \in U(P)}\{\%A\}) \subseteq Cn(P)$.

For the third fact, let any $A \in U(Q)$. Then $P \vdash Q \vdash \{\%A\}$, thus, by (%1) we get that $A \in U(P)$. Therefore, $U(Q) \subseteq U(P)$.

The fourth fact is implied by the second trivially.

The fifth is implied by the relation $P \cong Cn(P)$ and the fourth fact. □

Open Vocabulary Assumption

There is also another approach to the problem that motivated the introduction of the existence assertion operator and the CVA. This is expressed by the *Open Vocabulary Assumption* (OVA), which is the opposite of CVA: under this assumption, all kinds of bizarre elements that the DL namespace allows us to define are assumed to exist in the ontology. For most of them, no information may be available at the time, but this does not mean that they are not part of (or relevant to) our ontology. For example, a University ontology may contain no information about penguins, yet the concept PENGUIN is part of the ontology; the same holds for any other concept imaginable.

Therefore, an ontology under OVA corresponds to an ontology whose signature contains all the elements in the DL namespace. In this case, the signature structure is always equal to the DL namespace, so we lose no expressive power by dropping it. Moreover, addition and removal of namespace elements makes no sense under OVA, because such elements will always be part of the DL namespace (thus part of the signature of the ontology). For the same reason, the introduction of the existence assertion operator does not make sense under OVA. Notice that the OVA model is implicitly used in both classical logics and in the formal definition of DLs.

Discussion on OVA and CVA

OVA and CVA represent different viewpoints regarding the use of namespace elements in an ontology. CVA discriminates between relevant and non-relevant elements, while OVA considers all elements relevant. As already mentioned, an ontology under OVA corresponds to an ontology whose signature contains all the elements of the DL namespace; an ontology

under CVA corresponds to an ontology whose signature contains exactly the elements that explicitly appear in some axiom of the ontology. Such elements may appear in "standard" DL axioms (like $A \sqsubseteq B$) or in existence assertion axioms (like $\%A$). The existence assertion operator makes sense only under the CVA model.

The assumption (CVA or OVA) chosen for the ontology has several consequences in both reasoning and ontology evolution. It also affects the definition $\langle L,Cn \rangle$ of the underlying DL. Under CVA, we should include in L all axioms of the form $\%A$ for $A \in N_{DL}$; furthermore, the consequence operator is affected (due to the postulates (%1), (%2)). On the other hand, OVA requires no modifications to the standard definition of L and Cn, because the standard DL specification implicitly assumes the OVA model already.

Under CVA, the existence assertion operator plays a central role in reasoning: a namespace element A exists in an ontology (is part of its vocabulary) iff $\%A$ is a consequence of the ontology. If A does not appear in any axiom (including axioms of the form $\%A$), then $\%A$ is not a consequence of the ontology (by the semantics of %), thus A is not considered a part of (or relevant to) the ontology.

This fact has several strange side-effects; for example the axiom $A \sqsubseteq A$ is no longer a tautology. This is true because $A \sqsubseteq A$ implies $\%A$, so it implies that A exists in the ontology; but A does not exist in an ontology that does not contain A in any of its axioms, so $A \sqsubseteq A$ is not implied by any ontology (so it is not a tautology). If we accept $A \sqsubseteq A$ as a tautology, then every ontology would imply $A \sqsubseteq A$, which in turn implies that A exists in the ontology; this way, PENGUIN would exist in the University ontology, voiding the founding principle of the CVA. Under the standard viewpoint, this corresponds to the inability to reason about elements that don't appear in the ontology signature. For similar reasons, $A \sqsubseteq B \sqcup C$ is not a consequence of $A \sqsubseteq B$ alone; however, $A \sqsubseteq B \sqcup C$ is a consequence of the set $\{A \sqsubseteq B, \%C\}$.

In this respect, OVA exhibits a more classical behavior. Under OVA, $A \sqsubseteq A$ is a tautology, as A "exists" in all ontologies. For the same reason, $A \sqsubseteq B \sqcup C$ is implied by $A \sqsubseteq B$. In conclusion, OVA requires no changes to the formal definition $\langle L,Cn \rangle$ of the DL, as the formal specification of DLs implicitly adopts OVA, as already mentioned; CVA requires changes in both L and Cn, because of the inclusion of the existence assertion operator and its effects on reasoning, due to (%1), (%2).

Regarding ontology evolution, only the CVA can capture changes of the form "add concept PENGUIN to the ontology" or "remove concept PENGUIN from the ontology". For OVA, such statements don't make much sense, because PENGUIN will be a part of the ontology anyway. For CVA, such changes do make sense; they are also possible and can be expressed using the existence assertion operator, as explained before. Therefore, the deficiency that motivated the discrimination between OVA and CVA can be dealt with in one of the following two ways:

- We choose to use the CVA and enhance the underlying DL with the existence assertion operator. This approach is more consistent with what is usually the case in the ontology-related literature. Under this option, the information normally stored in the signature structure can be fully expressed in the form of DL axioms and the problematic changes are possible and are cleanly included in our framework. This approach solves the problem by enhancing the underlying language.
- We choose to use the OVA and keep the underlying DL as-is (no existence assertion axioms are allowed). This approach is more consistent with the standard logical model, as

well as with the standard DL definition. When using this option, the problematic changes are not possible, because they simply make no sense in the underlying language; the signature structure is neither necessary nor useful because all namespace elements are assumed to be relevant to the ontology anyway. This solves the problem by making it void, i.e., by making this kind of changes impossible, because their formulation and effects are not expressible in the underlying DL.

Notice that most approaches in the literature use a kind of hybrid approach on the subject. The use of signatures implicitly forces the use of the CVA; yet, the existence assertion operator is not introduced, because the use of the signature structure eliminates the need for this operator. Addition and deletion of namespace elements is allowed (using the signature structure implicitly). Clean, formal semantics on what such an action means is missing. As is common practice in the relevant research, only informal, intuitive descriptions of the process are given. Furthermore, the addition and removal of namespace elements affects the signature, while other types of changes affect the set of axioms of the ontology; this disallows a uniform treatment of both kinds of changes. Our proposal provides the semantics required for a uniform and formal treatment of this part of the problem of evolution.

In standard approaches, there is no reference to the consequences of CVA in reasoning, thus assuming standard reasoning. This approach contains the "a posteriori" assumption that the signature structure contains all the elements necessary for the deduction at hand. This fact directly contradicts the initial assumption of closed vocabulary. In our framework, the introduction of the existence assertion operator and the use of the logical representation of ontologies (using Tarski's model) provide the basis for the definition of the formal semantics of the reasoning process in the presence of ontological signatures (or, equivalently, CVA). To the author's knowledge, this is the first explicit discrimination of the two models.

In the rest of this section, it is irrelevant whether we use the OVA or the CVA, as long as the assumption chosen has been correctly reflected in the <L,Cn> definition of the representation language (DL). Unless otherwise mentioned, our comments and results will be applicable to both models.

Defining an Ontology Evolution Algorithm

Before proceeding, it is useful to provide a formal definition of what constitutes an *ontology evolution algorithm* (or operation). Our approach follows the path taken in [HS05a], which also coincides with the standard approach used in belief change; under this approach, an ontology evolution algorithm is a function, mapping an ontology and a change to an ontology. Notice that, in our model, both the ontologies and the changes are represented by sets of axioms of the underlying language; therefore, for a given underlying logic <L,Cn>, an ontology evolution algorithm can be formally defined as a function $OE:P(L)\times P(L)\rightarrow P(L)$, where P(L) is the powerset of L (also denoted by 2^L).

We have defined four different types of evolution operations, namely revision, contraction, update and erasure; therefore, a complete ontology evolution system should support all four, by implementing four ontology evolution functions, one per operation. All four of them should have the same signature ($P(L)\times P(L)\rightarrow P(L)$), but the different semantics of the four operations does not allow us to use the same function for all. An alternative to this definition would be to add a certain flag on the change that indicates its semantics, i.e., the type of operation required. This changes the signature of the ontology evolution function (because its domain should include the flag) and it allows the integration of all four functions into one. In

this work, we will use the former approach, because it avoids the further definition of a "flag". The operation involved in each case should be clear from the context.

In the rest of this dissertation, we will use infix notation for the ontology evolution functions. For example, an ontology contraction operation between an ontology O and a contracted expression X will be denoted by O−X.

Notice that the above definition does not constrain an ontology evolution algorithm in any way; any function with the proper range (P(L)×P(L)) and domain (P(L)) can be so classified. This means that the definition engulfs algorithms (i.e., functions) which exhibit properties undesirable for the purpose of ontology evolution. In the following, we will address some of the properties which affect the rationality of the result of an ontology evolution algorithm. These issues have already been addressed in the belief change literature, and were briefly reviewed in section 2.4, but will be now revisited under the prism of ontology evolution. As we will see, in most cases, the considerations that have appeared in the belief change literature are valid in the context of ontology evolution as well.

The Semantics of the Set of Changes

As explained in previous sections, a change request is represented using a set of propositions (i.e., axioms) of the underlying DL. In belief change, the change is usually represented using a single proposition. Our approach is, of course, more general than the standard belief change option; the question is, is this generalization appropriate or necessary for ontology evolution? We argue that the properties of the representation languages commonly used in ontologies (such as DLs) make such an option necessary.

Most belief change approaches assume that the underlying logic contains the usual operators of classical logic, like $\wedge$, $\vee$, $\neg$, $\rightarrow$ etc and includes classical tautological implication. Furthermore, sets of expressions have conjunctive semantics, so the set $\{A, \neg B\}$ is actually equivalent to $\{A \wedge \neg B\}$ (but see [KLM05] for an alternative semantics). This way, any finite set can be equivalently represented by a singular set, containing the conjunction of the finite set's propositions. These assumptions are taken for granted in most works of belief change, an indicative list being [AGM85], [Boch02a], [Del03], [Gro88], [Mak85], [Neb89]. Under these assumptions, generalizing the change to be a set of expressions (instead of a single one) is not a real improvement over the standard case, complicating the problem without adding any interesting generality to it. This is true because changes which are expressible *only* via infinite sets of expressions are not interesting in practice (because they cannot be represented in a finite machine), while the changes expressible via finite sets of expressions are also expressible by a single expression.

However, these assumptions fail for DLs; in many DLs, the conjunction of axioms is not possible, as the axioms have equational semantics (see [Bur98] for details on equational logic). For example, if the underlying DL is ALH, how could we express in a single axiom the set $\{A \sqsubseteq B, R \sqsubseteq S\}$ (under the standard conjunctive semantics), where A, B are concepts and R, S are roles? It follows that, for many DLs, there are facts which are expressible by a set of axioms, yet non-expressible by any single axiom. In this case the proposed generalization does make sense, as the new information (i.e., change) might not be expressible using a single axiom. For this reason, we believe it would be unnecessarily restrictive to constrain the change to be a single axiom only, as this does not take full advantage of the expressive power of the underlying DL.

Assuming that the change is a set of axioms does not address the problem of the semantics of this set of axioms. There are different ways that a set of propositions can be interpreted (e.g., [KLM05]). In this work, we assume the standard semantics for a set, i.e., that a set represents a group of axioms all of which should be satisfied (conjunctive semantics); this is the standard option for sets of axioms in DLs, as well as in most knowledge representation languages.

For belief change (and for ontology evolution), there are even more facets to this problem: a set of expressions may represent an iterated change [DP97], [Lib97], [Nay94]; or it may represent an entire new KB, in which case we are dealing with belief merging [Kon00]; in the case of a contraction operation, a set of expressions may have package or choice semantics [FH94]; alternatively, we may adopt the standard semantics, under which a set of expressions represents the conjunction of a group of beliefs that should be used to perform a certain operation (revision, contraction, update or erasure) upon the underlying KB. All these issues are equally applicable for ontology evolution and should be resolved before addressing the problem of determining the effects of the change, because the expected result of a certain change differs, depending on the semantics adopted.

The proper semantics chosen for each change is obviously depending on the application at hand. In current ontology evolution research, only the standard option is considered; the case of merging is dealt with in the fields of ontology merging and integration (under a different viewpoint) and the other options have not been considered. We believe that the standard option is the most suitable as an initial approach to ontology evolution, given the immaturity of the field; however, future research on the subject should address the other options as well, as these alternative semantics are useful in certain cases.

Foundations and Coherence Theories (Belief Bases and Belief Sets)

As was argued in section 2.4, there is a fundamental philosophical choice to be made in belief change regarding the representation of the knowledge, i.e., whether the explicitly represented knowledge serves as a justification for our beliefs (a belief base under the foundational semantics) or whether it simply forms a manageable representation of an infinite structure (a belief set under the coherence semantics).

This choice is very important, greatly affecting the ontology evolution (and belief change) algorithms considered. However, it is orthogonal to the representation language used, because it is related to the knowledge level [New82]. The choice of the viewpoint to employ depends on philosophical intuition, personal preference and on the intended use (application) of the KB (ontology in our context). Therefore, all the arguments, ideas and results discussed in the belief change literature (see, for example, [Gar92b], [Han92], [Han98]) are equally applicable here.

In current ontology evolution literature, the foundational theory is usually implicitly followed because the graphical representation of the ontology (containing only the explicit facts) dominates the decisions on what should be changed. No arguments are made in favor of this option, it's just followed without explicit mentioning or justification. In fact, the more practical character of ontology evolution as opposed to general belief change and the need for efficient ontology evolution algorithms favor this option. However, such a choice should not be taken lightly; only future research can verify that this option is indeed more appropriate for the development of rational ontology evolution algorithms. For more details on the pros and cons of each of the alternatives (in the belief change context), refer to [Gar92b].

Postulation Methods Versus Explicit Constructions

To the author's knowledge, all current approaches for ontology evolution use the explicit construction approach, by providing specific (manual or semi-automatic) algorithms to deal with the problem. This is due to the fact that a manual (or semi-automatic) approach presupposes user participation, which cannot be predicted or postulated; the explicit construction approach is partly justified by the more practical nature of ontology evolution, as opposed to general belief revision. However, it is generally acceptable that both approaches should be developed in parallel, one benefiting from the results of the other [Mak85].

In this respect, we believe there is a certain gap in current research, partly caused by the lack of efficient formalizations of the processes behind ontology evolution, partly by the participation of the ontology engineer in the evolution process and partly by the focus of current research on the technical and practical issues related to the problem. The formalization proposed in this thesis will hopefully motivate future postulation approaches. Furthermore, in subsequent chapters of this dissertation, we will describe our work on the application of the AGM theory [AGM85] to ontology evolution which introduces a certain postulation for ontology contraction, inspired by the AGM theory of contraction; this work is (to the author's knowledge) the first postulation attempt for the problem of ontology evolution.

Principle of Primacy of New Information

This principle has a special importance in the context of ontology evolution. The distributed nature of the Semantic Web and the many diverse uses of ontologies in agent-based systems or other autonomous environments, does not allow us to accept the new information in an unconditional manner in all applications. In such cases, *non-prioritized belief change* [Han99a] may prove useful, under which it is possible that part (or all) of the change is rejected (i.e., ignored), weakened, or otherwise modified to "fit" the existing knowledge. This approach applies when, for example, the source of information is untrustworthy. We believe that, in the context of the Semantic Web, the choice of the approach to be taken regarding this issue should be made on a per-application basis.

In current ontology evolution algorithms, the user selects the change to be made, which is then accepted unconditionally by the system, so the principle is generally valid [HS05a]. However, the active participation of the user in the change process allows him to select only a part of the change for application in the ontology. So, we could say that the principle is valid for the system itself, even though it can be implicitly invalidated by the user, if he chooses to modify an external change request before applying it to the system.

Principle of Irrelevance of Syntax

In the ontological context, the validity of this principle determines whether the syntactical formulation of the ontology or the change will affect the result of ontology evolution. Normally, semantic, rather than syntactic, considerations should be the driving force behind ontology evolution; in this sense the principle is valid in our context. Notice however that under the foundational viewpoint, which is usually implicitly assumed in ontologies, a stronger version of this principle may be more appropriate, like in standard belief change. In any case, we believe that the validity of this principle does not depend on the representation language itself, but on the nature of the change and the KB (i.e., the application); it is also affected by our choice regarding the knowledge model to use (coherence or foundational).

Consistency, Inconsistency, Coherence and Incoherence

The issue of ontology consistency (i.e., the Principle of Consistency Maintenance in the terminology of belief change) is very important for ontology evolution in general [HS05a] and, in particular, for the semantics of change phase; according to [HS04] and [SMSS03], the main goal of this phase is to ensure that the result of the change is a consistent ontology. The problem with this statement is that it is not clear what exactly a "consistent ontology" is; the term "consistent" (and "inconsistent") is another example of an overused term which has caused a certain amount of confusion in the literature.

For example, in [BCM99] and [HHHSS05], an ontology is called consistent iff it has at least one model; in the DL terminology, this means that there should be at least one interpretation that satisfies all the (DL) axioms of the ontology. In [MMSRV03], a consistent ontology is one that satisfies the set of invariants defined in the ontology model and all its used entities are defined. The definition is extended for sets of ontologies which have some kind of dependency and for replicated ontologies; these issues are outside the scope of our discussion. According to [HS04], consistent is an ontology which conforms to its "consistency model". Under this viewpoint, there are various notions of "consistency", each with its own "consistency model". In [HS05a], a similar use of the term is adopted, by defining three types of consistency, namely structural, logical and user-defined, which are based on certain "consistency conditions", depending on the underlying ontology model. Under [SMMS02], "inconsistency" may be semantical or syntactical. In [SMSS03], certain conditions that should be satisfied by any consistent ontology are presented, forming a "consistency model" in the terminology of [HS04]. In [PT05], the term "consistency model" is used again to determine consistent ontologies, but, in this case, the term refers to an ontology that conforms to the constraints imposed by the underlying ontology language.

Unfortunately, these different uses of the term are not compatible with each other. In this thesis, we will use two different terms to describe the various understandings of the term consistency (and inconsistency). *An ontology will be called inconsistent iff there is no interpretation satisfying all the DL axioms in the ontology; it will be called incoherent iff it does not satisfy certain predefined constraints or invariants related to efficient ontology design.* These constraints describe the ontology's "consistency model", in the terminology of [HS04]. Two popular sources of incoherence in this sense is the existence of concepts which are unsatisfiable, or the existence of roles which have no predefined range or domain. Inconsistency, in our sense, is defined along the lines of [BCM99] and [HHHSS05] and corresponds to "logical inconsistency" in the terminology of [HS05a].

Notice that an inconsistent ontology implies a logical contradiction, so any DL formula is inferable by such an ontology. An incoherent ontology implies bad design which could cause problems in the long run. Of course, the exact definition of "bad design" depends on the application at hand. We will discuss the differences between inconsistency and incoherence in more detail later in this subsection.

There are several causes for inconsistencies (and incoherencies) in an ontology; in [HHTGV04], four different scenarios which may cause such behavior in ontologies in the Semantic Web are identified. Several methods have been developed in the literature in order to deal with inconsistencies (and incoherences); in [HHHSS05], these different approaches are classified and evaluated and four different general methodologies for dealing with

inconsistent ontologies are identified. It is argued that these methodologies are applicable in different use cases and have slightly different input, output and general requirements.

The first general methodology that appears in [HHHSS05] is the repairing of inconsistent (or incoherent) ontologies; this repair involves the identification and removal of the problematic axioms so that an inconsistent (incoherent) ontology becomes consistent (coherent). A typical example of this methodology is [SC03], in which ontologies with unsatisfiable concepts are called incoherent and the process of identification and elimination of such concepts is called "debugging". Such ontologies are considered to contain "logical contradictions" and a debugging algorithm for such ontologies is proposed; this algorithm identifies the problematic concepts and provides clever algorithms for tracking down the sources of this problematic behavior. A similar approach and terminology is used in [FS05] and [WHRDS05]; the latter work focuses on OWL DL ontologies. The merging tool Chimaera [MFRW00] has certain diagnostic tools allowing one to check the ontology for several types of incoherencies (called inconsistencies in that paper). The same holds for other merging tools as well, like PROMPT [NM00]. Classical DL reasoners can also be used to detect inconsistencies or incoherencies in an ontology, but provide little support for resolving them [MLB05].

The second approach of dealing with inconsistencies does not attempt to resolve them, but tries to develop a mechanism that will allow reasoning in the presence of logical contradictions (inconsistencies). In [HHTGV04] it is argued that, due to the chaotic nature of the Semantic Web, inconsistency problems are unavoidable. For this reason, the development of a non-standard inference relation is proposed and the desired properties of such a relation are described [HHTGV05], [HHTGV04]. This inference relation allows meaningful answers to queries, even when the underlying ontology is inconsistent; thus, such a relation allows us to ignore the issue of inconsistency, or postpone the resolution of an inconsistency until a later time. This is similar to the approaches used in paraconsistent and nonmonotonic reasoning [Ant97] to deal with the explosive nature of classical entailment in the presence of contradictions. A similar approach appears in [HH00], where it is proposed that inconsistencies are avoided altogether by using a language (like SHOE) that disallows them.

The third approach is inspired by ontology versioning: the development of a new version of an ontology may render it "inconsistent" with respect to another ontology, application or agent that uses it, even if the new version itself is consistent; under this understanding, "consistency" refers to compatibility. To deal with this problem, different interoperable versions of the same ontology should be available, as well as a mechanism for reasoning and querying upon these different versions; this way, the ontology, application or agent that uses the problematic ontology could choose to use one of its previous versions instead of the most recent one, in order to avoid these compatibility problems. An example of such an approach is [HS05b], where a temporal logic is used to allow contemporaneous querying of different versions of the same ontology. Furthermore, any ontology versioning algorithm (e.g., [KF01], [KFKO02]) could be considered as a method for dealing with this type of "inconsistency".

The fourth (and most important for our purposes) methodology is consistent ontology evolution, which deals with the issue of avoiding inconsistencies during ontology evolution. Guaranteeing that the resulting, evolved ontology is consistent is one of the most important concerns during ontology evolution [HS04], [SMSS03], so most ontology evolution tools detect inconsistencies (and incoherencies) and present them to the user for resolution. This

requirement is also expressed by the Principle of Consistency Maintenance in the belief change literature, and it will be one of our main concerns in this thesis as well.

In this respect, we argue that ontology evolution tools need only concern with inconsistencies and ignore incoherences. Incoherences are probably signs of bad design but nothing more; tools that identify incoherences and provide methods for resolving them (like [SC03], [FS05], [WHRDS05], [KPS05]) are clearly desirable, but such a process has nothing to do with ontology evolution, being more related to ontology design. Therefore, such modeling problems should not be automatically corrected by an ontology evolution algorithm; instead, they should be simply pointed out to allow the knowledge engineer to handle them at his own leisure.

Inconsistent ontologies are undesirable in knowledge representation. An inconsistent ontology implies anything, so it will return irrational and conflicting answers to queries under a classical reasoner; this "explosive" behavior is problematic [HHTGV04], regardless of the application at hand, because such an ontology does not really tell us anything about the domain (as it implies everything). On the other hand, incoherent ontologies are not necessarily undesirable, even though in many cases an incoherent ontology is a potential source of contradiction that could lead to an inconsistent ontology.

We will explain our arguments with an example. Consider the well-known penguin example, in which the concept PENGUIN is defined as both a flying and a non-flying animal:

Example 3.3. Ontology: PENGUIN_ONTOLOGY

BIRD⊑FLY	(birds fly)
PENGUIN⊑BIRD	(penguins are birds)
PENGUIN⊑¬FLY	(penguins don't fly)

In several contexts (e.g., [HS04]) the PENGUIN_ONTOLOGY would be considered inconsistent; under our terminology, this ontology is only incoherent, but not inconsistent. It is incoherent because the concept PENGUIN is necessarily an empty concept under the given conceptualization and empty (i.e., unsatisfiable) concepts are undesirable in most contexts, being excluded from the ontology's "consistency model". The ontology is consistent though, because there are interpretations in which all the given DL axioms are satisfied. The problem with this ontology is that, once we add an assertion stating that a certain individual belongs to the concept PENGUIN (for example PENGUIN(PENGO)), the ontology becomes inconsistent; however, no inconsistency arises until such an assertion is made. This is the source of the problem and the reason that unsatisfiable concepts are usually undesirable: an unsatisfiable concept is, in a sense, a potential source of contradiction when we start populating the classes.

However, unsatisfiable concepts are not necessarily problematic. Let us change the previous ontology as follows: replace BIRD with HORSE, FLY with ¬HORNED and PENGUIN with UNICORN. The resulting ontology is:

Example 3.4. Ontology: UNICORN_ONTOLOGY

HORSE⊑¬HORNED	(horses don't have horns)
UNICORN⊑HORSE	(unicorns are horses)
UNICORN⊑HORNED	(unicorns have horns)

Performing the same analysis as before, we reach the conclusion that a unicorn both has and does not have horns. This is again an incoherent ontology, but not an inconsistent one. The concept UNICORN is unsatisfiable. However, could one characterize this behavior as

"problematic"? Probably not. To the author's knowledge there are no unicorns; in effect the concept UNICORN *should* be an empty one. It could even be the intention of the ontology engineer for the concept UNICORN to be unsatisfiable, to denote exactly the fact that unicorns do not exist.

Therefore, the problematic behavior of the PENGUIN_ONTOLOGY stems from our common knowledge that penguins in fact do exist, so an unsatisfiable concept by the name of PENGUIN is problematic; this argumentation collapses in the UNICORN_ONTOLOGY, exactly because our background knowledge tells us that unicorns do not exist. The important point here is that the two ontologies are structurally identical (see figure 3.2), so a computer-based system cannot discriminate between the two. Only a human, using a vast amount of background knowledge, can detect the difference between these two cases.

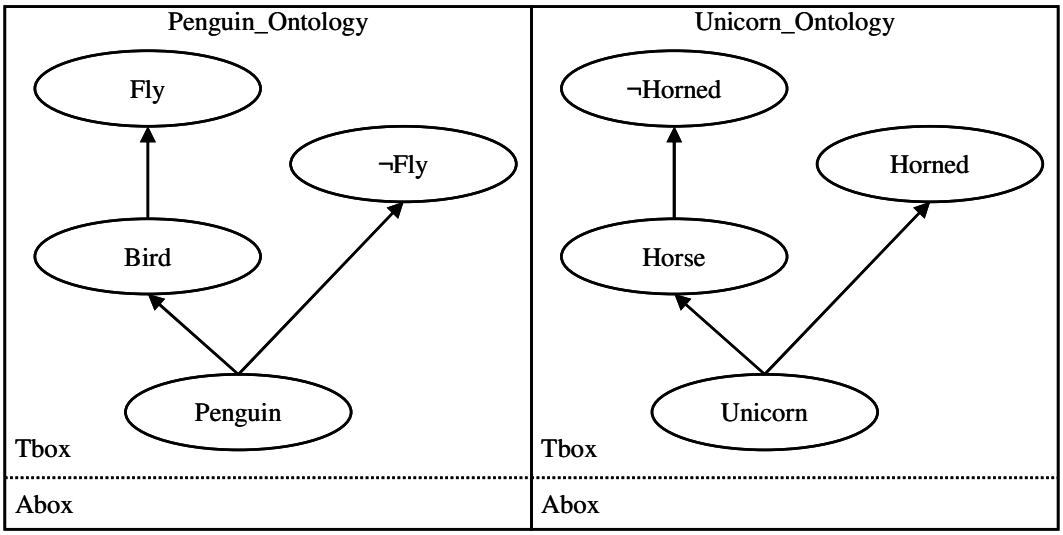

Figure 3.2. The PENGUIN_ONTOLOGY and the UNICORN_ONTOLOGY

Now consider an ontology with the facts:

BIRD⊑FLY (birds fly)

PENGUIN⊑BIRD (penguins are birds)

This ontology is both coherent and consistent. Assume that we revise this ontology with the fact:

PENGUIN⊑¬FLY (penguins don't fly)

Simply adding the new fact in the ontology would lead to incoherency (but not inconsistency). What should an ontology evolution algorithm do in this case? In many contexts, it is proposed that the algorithm should detect the incoherency and take an action against it, for example by removing or weakening the axiom BIRD⊑FLY [HHTGV04]. But, is that really what we want to do? Should we give up our belief that birds fly because of the fact that such a belief makes the class PENGUIN unsatisfiable? If you feel that the answer is positive, then consider the same revision performed in the unicorn example: should we give up our belief that horses do not have horns, just because someone came up with an imaginary concept called UNICORN? We believe not; at least not until some new knowledge asserts that the concept PENGUIN (or UNICORN) is in fact non-empty, in which case an action against this problematic behavior (inconsistency) is justified. But no such action is justified against an incoherence, because no computer system could tell the difference between a really problematic incoherence (i.e., PENGUIN) and a non-problematic "incoherence" (i.e., UNICORN). Similar arguments can be made for other forms of incoherencies, like the

existence of roles with no predefined range and/or domain [MMSRV03], classes with only one subclass [SMMS02], undefined entities [MMSRV03] or the explicit storage of redundant statements (i.e., axioms) [HS05a].

Our analysis shows that incoherencies are not directly related to the problem of ontology evolution. Spotting unsatisfiable concepts (as well as other types of incoherencies) is undoubtedly an important task, because incoherencies usually denote a problematic, or suboptimal ontology design [SC03]. If an incoherence is identified, it should be pointed out, warning the knowledge engineer of a potential design problem and allowing the continuous refinement of the ontology [SMMS02]; for this reason, tools that detect incoherences are highly desirable. However, the development of such tools is outside the scope of the ontology evolution field, being more related to the area of ontology design and quality of representation.

There is also another very important difference between incoherent and inconsistent ontologies: inconsistent ontologies are *always* undesirable, because an inconsistent ontology returns non-meaningful answers to queries, so the sources of inconsistency should be tracked down and eliminated during ontology evolution, ontology merging or any other process that created them; incoherent ontologies are *usually* undesirable, so the sources of incoherency should be pointed out for further processing that will determine whether they are indeed undesirable or not; yet, such an action need not be taken immediately, because an incoherent ontology (unlike an inconsistent one) can still provide meaningful (i.e., non-contradictory) answers to a user.

Principle of Adequacy of Representation

There is yet another type of "inconsistency" that is addressed in the ontology evolution literature: the term "structural consistency" refers to ontologies which are formulated according to their representation language, i.e., using no features, constructions or axioms which are not included in the language [HS05a]. This issue corresponds to the Principle of Adequacy of Representation in the belief change terminology, i.e., to the requirement that the new, evolved ontology (respectively KB) must be formulated using the same representational language as the original ontology (respectively KB).

We argue that this problem has nothing to do with inconsistency or incoherency; an ontology that does not conform to the requirements of the underlying representation language is simply ill-formed. It is the same as using the axiom $\forall A.B \sqsubseteq C$, for A a concept term: such an axiom does not denote an inconsistency; it's just not part of the underlying DL. Regardless of the term we use to describe this problem, an ontology evolution algorithm should always make sure that the resulting ontology is properly expressed according to the rules of the underlying representational language. This is a most basic prerequisite for any kind of formal or informal treatment of the problem. This basic requirement has already been incorporated in our definition of an ontology evolution function (i.e., algorithm) and we will always implicitly assume that it holds, without further mentioning.

Principle of Fairness

The Principle of Fairness is a very important, but underestimated principle. It guarantees the determinism and reproducibility of an ontology evolution (or belief change) algorithm. It is important because non-determinism may lead to complications, such as the inability to predict the result of a change or the inability to "undo" a certain change. Current ontology evolution

algorithms, being human-driven, are non-deterministic: one cannot predict the sequence of changes that a user will initiate in response to a certain need (i.e., a complex new fact), because not all users share the same views on how a certain change should be implemented [SMMS02]. On the other hand, the Principle of Fairness is an inherent property of our definition of an ontology evolution algorithm, because a function is by definition deterministic. This is also one of the advantages of our methodology: automatic ontology evolution guarantees that the result of a change will be reproducible, predictable and reversible.

Principle of Minimal Change

As already stated, the Principle of Minimal Change is the most important issue to consider before developing a rational belief change algorithm. Any given change can be resolved in several ways; the system (belief change algorithm) should choose the one that minimizes a certain undesired quantity (e.g., information loss) or maximizes a certain desired quantity (e.g., reliance in the resulting KB). There are several ways to count these quantities, each resulting from a different metric and leading to a different implementation of this principle; such an implementation underlies every work on belief change, determining (to a large extent) its properties and behavior when faced with a change.

This is also true in the ontology evolution context: according to [HS05a], the result of an ontology evolution operation should be an ontology that satisfies the user's requirement for a change (i.e., the Principle of Primacy of New Information) and it is at the same time a consistent ontology (i.e., satisfies the Principle of Consistency Maintenance); since there are several different ways to satisfy these requirements (just like in belief change), one should choose the one that modifies the original ontology in a minimal manner [HS05a]. This is the Principle of Minimal Impact, in the terminology of [HS05a] (i.e., the Principle of Minimal Change in the belief change terminology). Notice the closeness between the considerations expressed in [HS05a] and the ones that have been expressed in the context of belief change (using a slightly different terminology) in [Gar92a]; this is another argument in favor of the connection between the two research areas.

Under our axiom-based viewpoint, the basic "block of knowledge" in ontologies is the axioms. Therefore, the loss of information could be counted in terms of the number and importance of the axioms that need to be removed from the ontology in order for the new information to be accommodated; alternatively, the loss of information could be counted in model-theoretic terms (via some kind of distance metric between the interpretations satisfying the original and the modified ontology); or through some specially designed distance metric between ontologies; or via certain conditions (postulates) that identify acceptable and non-acceptable transitions. The counterparts of all these methods have been successfully used in the belief change context (see [AGM85], [Dal88b], [GM88], [Gro88], [Han96], [KM90b] for some examples), greatly simplifying our task.

Current approaches to ontology evolution focus on the structure of the ontology in a graph-based representation, so there is no similar "basic block of knowledge". The concepts, roles and individuals are the building blocks in this case, as well as the connections (relations) between these objects. Since axioms are usually not considered to be parts of an ontology [NM99a], the graphical representation could prove useful in determining the distance between the original and the modified ontology in a formal manner; this way, the Principle of Minimal

Change could be implemented by minimizing this distance. In this respect, the study of specially designed distances between graphs could prove useful.

Current ontology evolution tools avoid the difficulty of defining such a distance through the use of semi-automatic evolution methods: given a change, such tools usually provide the knowledge engineer with certain alternatives, allowing him to select the one that minimizes the "information loss". Since the selection is made by the user, he is responsible for minimizing the information loss resulting from the change. For example, when a concept is removed, its former instances could be either removed or reclassified; it is up to the user to determine the best alternative, either directly, or indirectly, via some kind of evolution strategy [SMMS02]. In addition, hybrid approaches should be possible, for example removing some of the instances and reclassifying the others. Of course, there is no guarantee that these are the only alternatives, since there is no formal study showing that; also, there is no formal study (or metric) measuring the information loss of each of the alternatives. It is up to the user to select the less plausible facts for removal (or modification), based on his expertise on the domain and the ontology.

In [HS05a], a method based on the notion of confidence is sketched, but not validated. The authors prefer to use a simpler approach, using the notion of "structural connection" between axioms and counting information loss as being equal to the number of removed axioms. The alternative method (based on confidence) would exploit extra-logical information regarding our confidence on each axiom, allowing a computer system to detect and remove the less plausible axioms automatically. The notion of confidence is similar to the belief change notion of epistemic entrenchment [GM88]. Modifying an ontology using any of the above methods allows us to perform the change with minimal loss of information. However, the loss of information is counted differently in each approach, giving different meaning to the term "minimal"; thus, the result of the change would also be different in each case.

Example of Application

In this subsection we will sum up the most important features and ideas presented so far, by showing the practical feasibility of our approach through a simple, but intuitive example, which will be illustrated using both logical and graphical representations. Of course, the example will be addressed in an informal manner, as our proposal has not provided (so far) an explicit algorithm for ontology evolution but a set of general suggestions showing how the intuitions and results developed in the belief change literature can be exploited to produce working ontology evolution algorithms. More formal methods of dealing with changes will be presented in the following chapters of this dissertation. Despite that, we will provide hints on how the various problems that emerge during our analysis could be formally solved.

Let us consider a very simple ontology, asserting that elephants cannot fly (see the original ontology in figure 3.3):

Example 3.5. Ontology: DUMBO_ONTOLOGY

ELEPHANT$\sqsubseteq\neg$FLY (elephants don't fly)

Now, reading the story by Helen Aberson Mayer and Harold Perl regarding Dumbo, the little flying elephant, we learn that there is a certain individual, DUMBO, which is an elephant. So we revise our beliefs with the fact:

ELEPHANT(DUMBO) (DUMBO is an elephant)

This can be easily accomplished by adding this axiom to our original ontology, because this would result to no contradiction (first revision in figure 3.3). It is generally acknowledged in the belief change literature that revision should be the same as expansion when no contradiction occurs [AGM85], [Gar92a]. This is also true in the ontological context; for this reason, the problem of ontology revision is interesting only when the new information contradicts the old. This case occurs when, during our reading of the tale on Dumbo, we find out that Dumbo can actually fly. Our ontology should be revised with a set containing the following fact:

FLY(DUMBO) (DUMBO can fly)

Simply adding this fact to our ontology would result to an inconsistency (phase 0 in figure 3.3), because a certain individual (DUMBO) would be asserted to be a member of two concepts which are by definition disjoint (FLY and ¬FLY). The question is, how should we deal with this problem? In other words, how should we revise the ontology:

$$\{\text{ELEPHANT} \sqsubseteq \neg \text{FLY}, \text{ELEPHANT}(\text{DUMBO})\}$$

in the face of the new data:

$$\{\text{FLY}(\text{DUMBO})\}$$

One possible approach is based on the fact that the DL axioms used in this example can be translated to FOL [BCMNPS02]; following this translation, we could use one of the numerous belief change algorithms that deal with revision in classical logic and then translate the result back to a DL axiom. This approach is rather naïve, for two reasons: first, even though the majority of DLs are fragments of FOL, there are several ones which go beyond FOL expressiveness [Bor96]; second, even if the underlying DL is a fragment of FOL, there is no guarantee that the result of the revision will be expressible in this DL. If the result of the applied algorithm is a FOL proposition which is not expressible in the underlying DL, we would violate one of the most important principles of ontology evolution (and belief change), namely the Principle of Adequacy of Representation.

For these reasons, we would like to use an algorithm that will be applicable to DL axioms directly, without the need for any intermediate translation. In addition, we would like the system to be able to decide how to accommodate the new fact (DUMBO can fly), without human intervention; we believe that this is a plausible goal, because in standard belief change this problem has been successfully addressed.

In standard ontology evolution approaches a change request includes information on how the change should be resolved because the changes represent operations such as “Remove_Concept”, “Add_IsA”, “Move_SubTree” etc; this greatly simplifies the problem, but relies on human intervention to determine the type and order of these operations in response to the new data. In our framework, the information that DUMBO can fly is not directly helpful in determining the required modifications upon the ontology because it gives us no clue as to how to include this information to our ontology; instead, the system should determine the required actions.

Following the general principles of belief revision, the proper approach towards implementing this revision is to find a weakening of the original ontology:

$$\{\text{ELEPHANT} \sqsubseteq \neg \text{FLY}, \text{ELEPHANT}(\text{DUMBO})\}$$

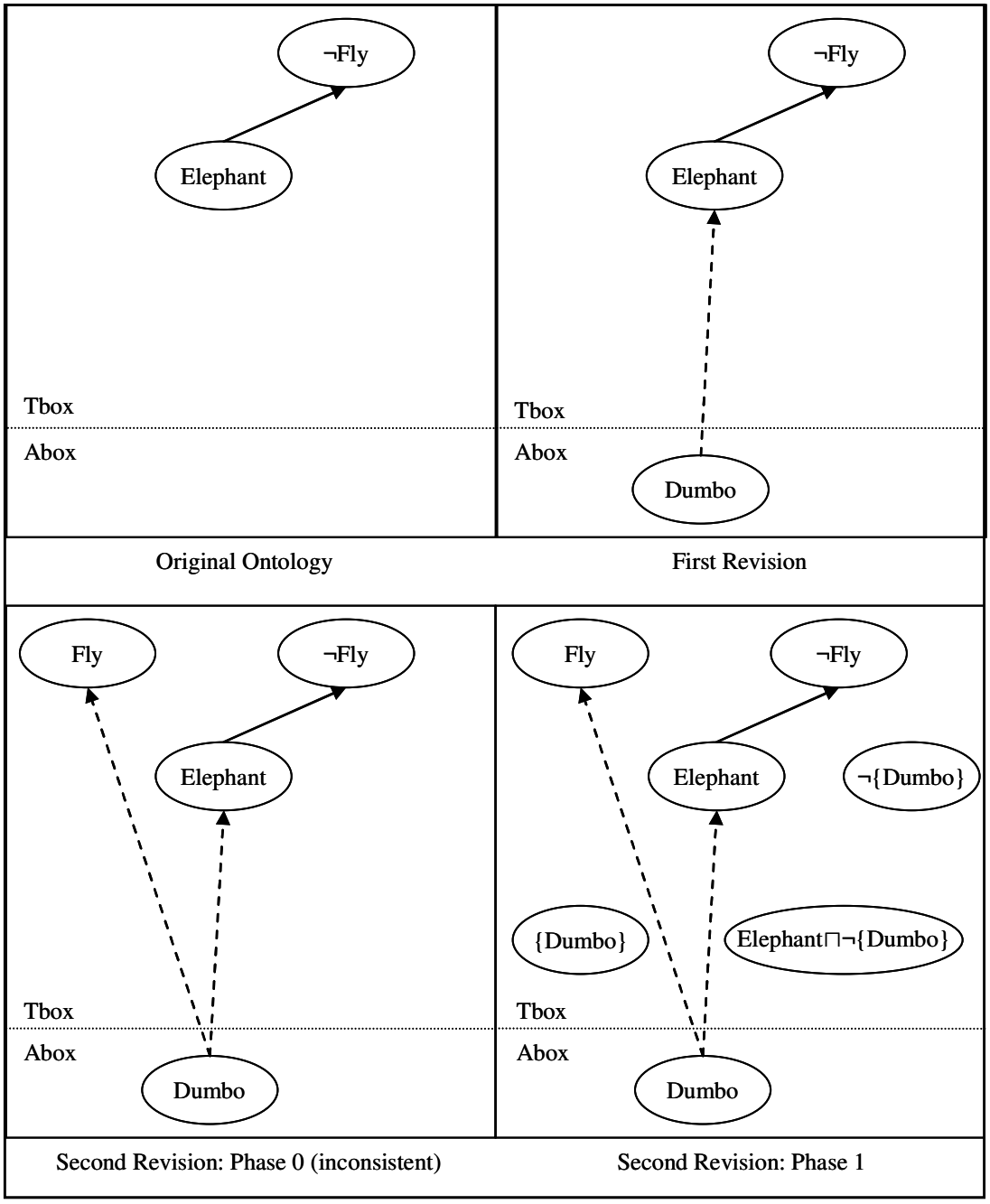

Figure 3.3. Example of a revision (DUMBO_ONTOLOGY) (continued in the next page)

This weakening should be such that the addition of the revising expression to it will not lead to an inconsistency. This weakening should be "minimal" in the sense of the Principle of Minimal Change; what remains to be fixed is the definition of the term "minimal".

A human agent could in this case determine that, since the information came from a fairy tale, we should not treat it as a standard, but as an exception. Thus, we have no reason to eliminate our belief that elephants don't fly; instead, we should add an exception to this belief stating that all elephants, but DUMBO, cannot fly. This conclusion is based on common sense (e.g., fairy tales are not reliable) and on extra-logical information (e.g., reliability of each of our beliefs). An alternative approach would be to weaken (or remove) the fact that DUMBO is an elephant, because the resulting contradiction caused us to simply lose our faith in what we read in the fairy tale.

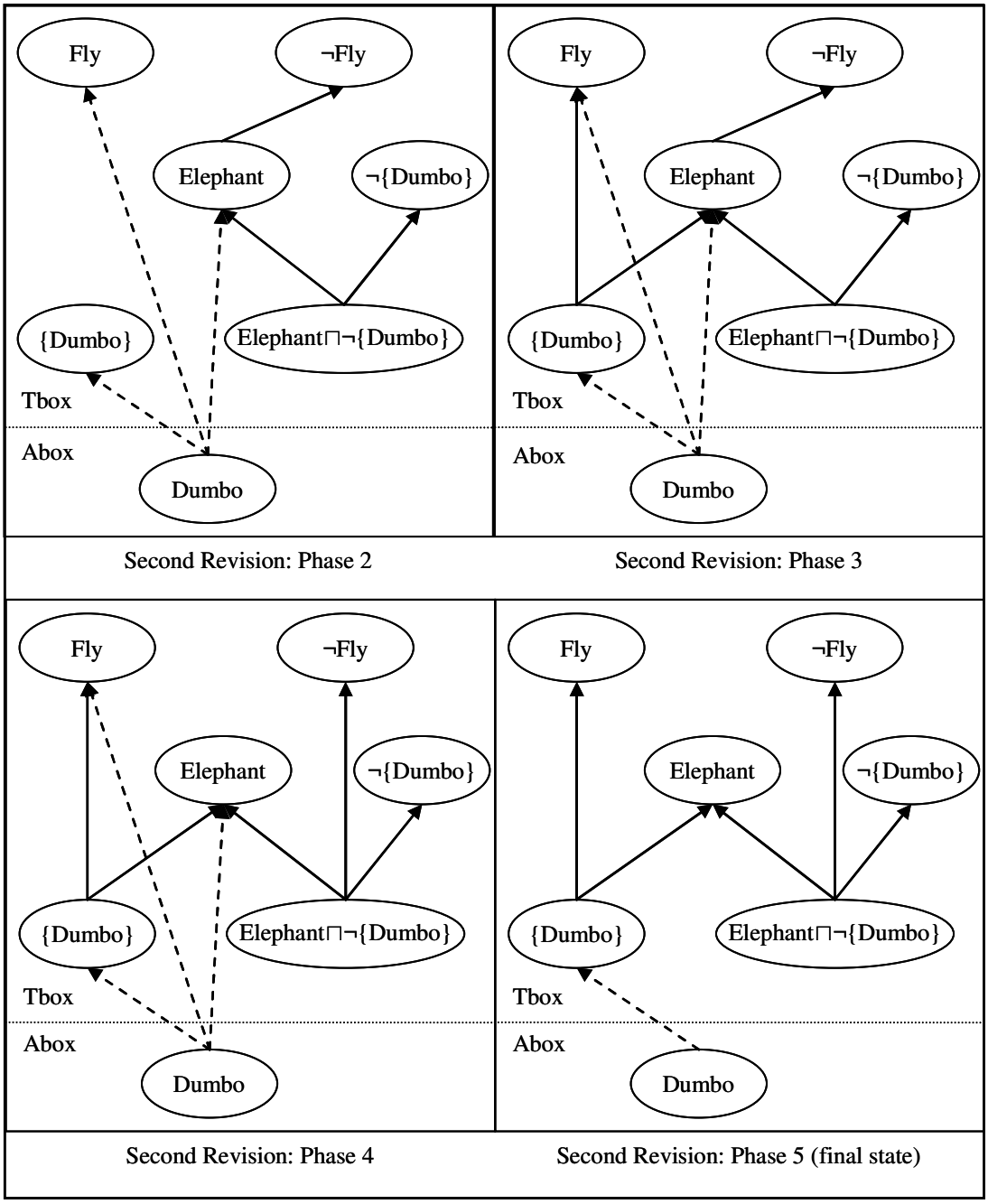

Figure 3.3. Example of a revision (DUMBO_ONTOLOGY)

In any case, this is something that the system should be able to decide upon. To allow this, we could for example equip our axioms with a certain entrenchment [GM88] or confidence [HS05a] ordering, which would determine the less reliable facts of the ontology. This option is not as simple as it may seem, because this ordering should be also extended to implicit axioms (i.e., consequences of the ontology) in a proper way (see [AM85], [GM88] for details). Alternatively, we could consider all axioms equally reliable (using no additional information) and use a certain algorithm that weakens the whole ontology uniformly, until the desired level of weakening is reached; examples of belief change algorithms that implement this idea are: [AGM85], [Dal88b], [Gro88]. Of course, these works are not directly applicable to DLs, as the underlying assumptions are generally different; however, the intuitions behind the algorithms are valid in our context.

Let us assume that the system chooses to weaken the axiom ELEPHANT⊑¬FLY. One extreme approach would be to eliminate the axiom altogether; we consider this a special case of weakening, in which the axiom is replaced by a tautology. However, as explained before, a better option would be to treat DUMBO as an exception. Under this viewpoint, we would like to replace the chosen axiom with the axiom: ELEPHANT⊓¬{DUMBO}⊑¬FLY. This would result to the following (revised) ontology:

Ontology: DUMBO_ONTOLOGY_REVISED

ELEPHANT⊓¬{DUMBO}⊑¬FLY	(all elephants, but DUMBO, don't fly)
ELEPHANT(DUMBO)	(DUMBO is an elephant)
FLY(DUMBO)	(DUMBO can fly)

This is a consistent ontology (Principle of Consistency Maintenance); the revision has resulted in minimal loss of information (Principle of Minimal Change) and has accommodated the revision (Principle of Primacy of New Information). We see that this result is therefore compatible with the most important general principles of the dynamics of change, as studied in belief change. There is one question that remains unanswered though: how could the system select the proper weakening of the ontology? Assuming that the axiom(s) to be weakened have been selected using one of the aforementioned methods, how would the system decide on the exact weakened version of those axioms?

This task could be accomplished either in model-theoretic terms, or using a certain syntactic or semantic scheme that iteratively weakens the selected axiom(s) until the desired level of weakening has been reached. Both approaches have been used in the belief change literature (e.g., [AGM85], [BKBW04], [Dal88a], [KM90b], [Mak85], [Wil97]). The problem has been addressed in the DL context as well [HS05a]. In addition, the scheme proposed for reasoning under inconsistency in [HHTGV05], [HHTGV04] could be adapted to address the same problem.

Figure 3.3 shows how this revision can be performed using current ontology evolution systems. Notice the number of operations upon the ontology graph that should be initiated in order to perform this (seemingly simple) change in the ontology:

1) Initially, (phase 1), we should introduce three new concepts ({DUMBO}, ¬{DUMBO}, ELEPHANT⊓¬{DUMBO}); these are necessary for the proper representation of the axiom ELEPHANT⊓¬{DUMBO}⊑¬FLY during subsequent phases.
2) During phase 2, we should add the obvious relationships between these concepts, by asserting that the individual DUMBO is an instance of {DUMBO} and connecting the concept ELEPHANT⊓¬{DUMBO} via IsA relationships to its immediate parents (ELEPHANT and ¬{DUMBO}).
3) Then, we should add two additional obvious IsA relationships between {DUMBO} and ELEPHANT, as well as between {DUMBO} and FLY (phase 3).
4) The IsA between ELEPHANT and ¬FLY should be removed and replaced by the weaker IsA between ELEPHANT⊓¬{DUMBO} and ¬FLY during phase 4. Notice that this is the main step of the evolution process; the previous and next phases were necessary in order to support phase 4.
5) Finally, we should remove the redundant membership assertions between DUMBO and ELEPHANT and between DUMBO and FLY, as these are direct consequences of other relationships in the ontology (phase 5); eliminating redundant relationships is considered necessary in several ontology evolution works, like [HS05a], [SMMS02]. This

elimination also makes the visualization of the ontology clearer, without altering the logical force of the ontology.

Notice that in certain contexts, the existence of a single subconcept of a given concept (as is the case with ¬FLY and ¬{DUMBO} in our example) is considered suboptimal behavior [SMMS02], [SMSS03], which should initiate further changes and refinements in the concept hierarchy. We don't address this issue here because we believe that this problem could be considered a mild type of incoherency; as argued before, dealing with incoherencies is part of ontology design (not ontology evolution), thus outside the scope of this thesis.

The considerations presented here are equally true regardless of whether the change refers to an individual, concept or role. As already mentioned, our viewing of knowledge as being expressible at the level of axioms allows a unifying approach to a change, regardless of whether it refers to an individual, concept or role, as all these "types of knowledge" are expressible in a uniform manner, using adequate axioms.

Furthermore, these considerations are partially true in the ontology merging context as well. Indeed:

Example 3.6. Consider the following ontologies, which are homogeneous and they use the same terminology and naming conventions, the same underlying DL language etc:

Ontology: TWEETY_ONTOLOGY1	
BIRD⊑FEATHERS	(all birds have feathers)
FEATHERS(TWEETY)	(TWEETY has feathers)
BIRD(TWEETY)	(TWEETY is a bird)
¬FLY(TWEETY)	(TWEETY can't fly)
Ontology: TWEETY_ONTOLOGY2	
BIRD⊑FLY	(all birds fly)
BIRD(TWEETY)	(TWEETY is a bird)
FEATHERS(TWEETY)	(TWEETY has feathers)

Both ontologies are consistent (and coherent); however, if one tries to merge them by simply taking the union of their axioms, the result would be an inconsistent ontology, because the individual TWEETY would be asserted to be both a flying and a non-flying individual. To address this problem, one could choose to weaken one or both ontologies before taking the union; the dynamics of this weakening are generally similar to the standard case (ontology revision). However, in the context of merging, it makes sense to consider the axioms that appear in both ontologies as more reliable than those that appear in one of them only [KPP05]; also, the Principle of Primacy of New Information is not applicable here.

In short, the general considerations (regarding ontology evolution) that appeared in previous subsections make sense and can be applied in (some part of) ontology merging, with the necessary adaptations to address the slightly different nature of the problem. Notice however that, in ontology merging, resolving any possible heterogeneities between the source ontologies is an important issue, whose resolution should precede any attempt of merging the ontologies. In this particular example, no heterogeneity issues were present. Therefore, the general principles of our work are applicable in the part of merging that resolves the resulting inconsistencies and presuppose that all heterogeneity issues have been somehow resolved.

3.4. Discussion and Comparison of the Approaches

We believe that our proposal will be helpful in resolving some of the deficiencies of current ontology evolution methods. As already mentioned, this approach is complementary (and not rivalrous) to the current state-of-the-art techniques, allowing them to resolve certain issues that are improperly handled, the most important ones being the manual or semi-automatic way of handling the changes, the immature nature of the field, the lack of adequate formalizations and the great number of different operations that are addressed in a non-uniform manner, while, at the same time, taking advantage of the very useful features that are being employed by such state-of-the-art tools, like transactional properties, undo/redo capabilities, intuitive interfaces etc.

In current ontology evolution research, the change is considered a one-off process; as such, its manual handling is feasible (even though not practical), as the ontology engineer can take the time required to perform the needed changes. Our method is addressed to applications in which changes occur often, so the ontology engineer cannot handle the burden of updating the ontology manually; it is also applicable in cases where it is difficult, impossible or undesirable for an ontology engineer to handle the change himself (for example in autonomous robots or software agents, in large ontologies developed by a group of engineers or in time-critical applications). In such cases, user-driven ontology evolution is not practical, so a framework allowing automatic ontology evolution is highly desirable.

Apart from applicability, we consider our method superior because it provides clean semantics by allowing a formal and uniform treatment to the problem of ontology evolution; this kind of formality is generally missing in standard approaches. It allows us to use the postulation approach to address the general problems of ontology evolution under a theoretical viewpoint (also missing in the current state of the art). In this respect, the issues addressed in the belief change literature may prove extremely useful towards the formalization and postulation of the process of ontology evolution; several examples of belief change concepts which are useful in the ontology evolution context have appeared in the previous section.

The formality of our approach also allows us to abstract from the knowledge representation formalism used to represent the ontology, using only the properties necessary to establish our desired results; on the other hand, existing ontology evolution methods heavily depend on the exact properties of the underlying ontology language [HS04], so they are not easily transferable to other formalisms (even to similar DLs). Finally, this formality led to the development of a novel formal distinction between CVA and OVA, a distinction that has been neglected in the relevant literature, as the issues involved in this distinction are dealt with implicitly, in a rather blurred and inconsistent manner.

Standard methods deal with each different atomic or composite operation straightforwardly. There is a certain matching, assigning each type of change (i.e., operation) with a certain process that describes the actions taken to address the given change. The price to pay for this simplicity is scalability: one new process must be defined for each new type of operation devised. Unfortunately, there is a great (in fact, infinite) number of different operations to consider [KN03], [SK03], [PT05] so the lack of scalability is a real bottleneck for future research. On the contrary, under our viewpoint, there are only four different operations to consider (namely revision, contraction, update and erasure); these engulf all the types of modifications that one may wish to perform upon an ontology, even though there is

no exact mapping between these four operations and the (atomic or composite) operations used in the standard approach.

Another important innovation associated with our approach is that it focuses on axioms, rather than on the structural connection between concepts, roles and individuals in a representation graph, as is usually the case in standard approaches. This axiom-based viewpoint allows us to use the full power of the underlying logical formalism; more importantly, it provides cleaner and more formal semantics compared to the standard approaches, which build on frame-like or object models [HS05a].

These observations show that our approach provides the ontology evolution community with tools, ideas and features that were before unavailable. This method, coupled with the exciting features provided by current ontology evolution tools, such as undo/redo capabilities, visual representation, collaborative editing, atomicity and transactional nature of changes, change propagation, verification features, inconsistency and incoherency detection tools, ontology repair capabilities etc, will provide interesting solutions to problems currently faced by ontology-based applications when modifications in the underlying ontology are required.

As stated in the introduction, the focus of this chapter was not to provide concrete answers and solutions to the various problems faced by ontology evolution researchers, but to set stable foundations upon which future work towards this objective may be based. In this chapter, our approach was justified and general guidelines towards the proposed research direction (namely the application of belief change techniques to ontology evolution) were provided and evaluated; one possible application of these ideas towards a definition of an ontology evolution algorithm will appear in subsequent chapters.

3.5. Similar Approaches

Introduction

There are certain recent works (at a preliminary stage) that apply belief change techniques to ontology evolution [KL04], [LM04], [MLB05]; these works were developed independently, but they closely follow our line of thought as described in this chapter. We will briefly present these approaches, as they are exemplary of the research direction we are proposing regarding ontology evolution. They can be viewed as applications of our viewpoint for certain belief change techniques.

AGM Theory in Ontology Evolution

One such work, at a preliminary stage, is [KL04], where the authors propose the use of the AGM theory [AGM85] for ontology evolution. Only some informal ideas regarding the connection of the AGM theory with ontology evolution are provided. The focus lies on the operations of contraction and revision and, following the lead of the AGM postulates, certain properties that should hold in a rational contraction and revision operation are presented. However, these properties are not directly applicable to many DLs, for the same reasons that the AGM theory itself is not directly applicable to such DLs (see [FPA04b]), so this is a problem that remains to be solved by the authors of [KL04].

In the next chapters of this dissertation, we will perform a similar study on the AGM theory and explain why the AGM postulates cannot be applied in the ontology evolution context as-is; we will also provide a generalization of the AGM theory, which is one possible

solution to the problems faced by the authors of [KL04] during their preliminary attempt with the AGM theory.

Epistemic Entrenchment in Ontology Evolution

In [LM04], the focus is on a certain preference ordering that is inspired by previous work on epistemic entrenchment [GM88] in the belief change literature. The interrelationship between the Abox and the Tbox of the DL KB in the context of ontology evolution is studied and the distinction between the two parts of the KB is taken seriously. Only revision is considered, studying how new knowledge can be incorporated in a DL KB without introducing inconsistencies. Unlike our approach, the new knowledge (change) is constrained to be a single DL axiom. The DL considered is ALU (only), while in this dissertation the family of DLs as a whole, as well as OWL, are considered. The effects of that work on ontology merging are briefly discussed, as well as the differences of this problem with classical belief merging [Kon00], [KPP05]. The authors also study ontology revision in a sense slightly different than the one used in this dissertation: under their understanding, the term ontology revision refers to the addition of an Abox assertion to the DL KB.

Maxi-adjustment in Ontology Evolution

Another interesting variation of a belief change algorithm appears in [MLB05], where the authors attempt to recast the maxi-adjustment algorithm [BKBW04], originally introduced for propositional knowledge integration, to the context of DLs. The maxi-adjustment algorithm allows the elimination of any inconsistencies that could arise in a stratified KB after its expansion with a new proposition; this process can, in turn, lead to the development of a revision algorithm. The authors propose a basic and a refined algorithm which apply the ideas of maxi-adjustment to DLs. Unfortunately, for their method to be applicable, the disjunction of DL axioms needs to be expressible, which is not the case in standard DLs. For this reason, the authors provide the necessary definitions and semantics for the introduction of disjunctive DLs, a certain DL extension for which axiom disjunction is allowed. This, however, constitutes a departure from the classical DL model, limiting the applicability of their approach.

Chapter 4: Generalizing the AGM Theory

If Edison had a needle to find in a haystack, he would proceed at once with the diligence of the bee to examine straw after straw until he found the object of his search... I was a sorry witness of such doings, knowing that a little theory and calculation would have saved him ninety per cent of his labor.

Nikola Tesla

4.1. Introduction

In chapters 4 and 5 we will attempt to apply a particular belief change theory to the problem of ontology evolution; this approach can be viewed as an application of the framework we proposed in chapter 3. Inspired by the notion of rational contractions as was formally defined by AGM in their seminal paper [AGM85], we will attempt to apply the same notion to DLs and OWL and determine whether it is possible to recast the original formulation of the AGM postulates so as to be applicable to these logics.

This discussion is split in two major parts. In the first part, we will deal with the generalization of the AGM theory in logics that can be described using Tarski's model; this part of our work can be applied to all logics under this framework, including DLs, OWL, as well as logics which are not used for the representation of ontologies. In the second part, we will focus on DLs and OWL only; using the results developed in the first part, we will provide more specialized results which are only applicable to these languages. The current chapter deals with the first part only; the second part will be presented in chapter 5.

4.2. Motivation and Goals

As already mentioned, the AGM model is the dominating paradigm in the area of belief change. Unfortunately though, the assumptions upon which the AGM theory is based disallow its application to certain interesting classes of logics. To our knowledge, there has been no attempt (yet) to extend this theory beyond its original scope. This part of the dissertation attempts to fill this gap by relaxing the original AGM assumptions and studying the feasibility of applying the AGM theory to a generic class of logics with only minimal assumptions as to their properties (i.e., to the logics under Tarski's framework).

The AGM theory was chosen because of its unparalleled influence in the field of belief change; we are hoping that it will have a similar impact in the field of ontology evolution as well. Our work focuses on the AGM theory of contraction; contraction was chosen because,

according to AGM, it is the most fundamental among the three belief change operators [AGM85], [Gar92a]. Moreover, the theoretical importance of contraction has been accepted by most researchers, even though revision is more often used in practical applications. Some preliminary results regarding the revision operator will be presented in section 4.9.

This part of our work is of dual significance: first, it determines the applicability of the AGM theory in several interesting logics which were originally excluded from the AGM-related research. Second, it forms a running example showing how belief change techniques can be applied to logics used for the representation of ontologies; this is important, because the problems encountered in our work are exemplary of the complications that could arise during the migration of other belief change techniques to ontology evolution.

As was shown by our brief review of the AGM theory in section 2.5, the intuitions that led to the development of the AGM theory are independent of the actual language used to represent the knowledge. This supports our belief that the concepts used in belief change (in this case the concept of rationality of a belief change operator) are independent of the underlying knowledge representation scheme. Since the AGM theory formally encodes common intuition about the properties that a rational belief change operator should satisfy, we believe that it would be of interest to apply the AGM model in any logic in order to determine the rationality of any given revision or contraction operator.

The main problem with this approach is that the formulation of the AGM postulates themselves uses the assumptions made by AGM in the formulation of their theory. Thus, the AGM postulates do depend on the underlying language, placing the work mainly in the context of classical logic and disallowing its direct application to several languages, including DLs and OWL which are the focus of this thesis. Indeed, a DL is not necessarily closed under the usual operators ($\neg$, $\wedge$, $\vee$, $\rightarrow$ etc), because DL axioms are of equational form (e.g., $A \sqcap B \sqsubseteq C$), so, for example, the negation of a DL axiom cannot be defined in general. The same holds for OWL, as well as for many other families of knowledge representation languages.

This issue is reminiscent of the connection between the problem of belief change with ontology evolution: the differences are not on the underlying intuitions, but on the representation languages and formalisms used. For the AGM theory, since the problem actually lies on the formulation of this theory instead of its underlying intuition, we feel that it makes sense to recast it in a setting general enough to contain DLs. As we will see, this generalization will uncover interesting results regarding the connection of the AGM theory with the foundational model [Gar92b] and lattice theory [Gra71], as well as a new representation result for the AGM postulates.

4.3. The Generalization of the AGM Theory

Basic Setting

As already mentioned, we will assume that a logic is described by an <L,Cn> pair. This framework is very general and includes all monotonic DLs, OWL, PL, FOL, DAML+OIL, RDF, modal logics, equational logics and many others. More importantly, it includes all the logics that satisfy the original AGM assumptions [AGM85]; thus, it is more general than the AGM framework.

The first step of our generalization is to generalize the definition of a contraction (and revision) operator to capture the more varying semantics of the wider class of logics that we consider. The original AGM definition constrains the operands of a contraction (and revision) operator to be a theory and a single proposition of the underlying language (i.e., $K'=K-x$ or $K'=K+x$, respectively).

Our first generalization in this respect is to assume that the first operand (the KB) is not necessarily a theory but can be any set of propositions of the underlying language. The reasons for this generalization are purely technical; this is made just for representation purposes. We still follow the coherence model upon which the AGM theory was based. The set is assumed to have the full power of its theory, i.e., its implications are important for the formulation of the result. It would be equivalent to assume that we calculate the logical implications of this set as a pre-processing step before contraction and revision is performed.

The second, and more important, generalization is to assume that the second operand (the change) can be any set of propositions. This is an essential generalization. In section 3.3, we explained why the change in ontology evolution should be a set of DL axioms, instead of a single axiom; the same considerations apply here.

Under these generalizations, both contraction and revision operators map two sets of propositions (the KB and the change) to another set (the contracted or revised KB). Thus, the contraction operator is a function $-:P(L)\times P(L)\rightarrow P(L)$; the same goes for revision. Notice that this is the same pattern that was used for the definition of an ontology evolution operator. Also, it should be clear that these sets have the standard, conjunctive semantics, so, for contraction we have choice semantics [FH94].

Generalized Basic AGM Postulates for Contraction

Having set our general framework, the next logical step of our generalization was to reformulate the basic AGM postulates of contraction (revision will be dealt with at a later point) in such a way as to be applicable to all logics of our more general framework, while preserving the original intuition that led to each postulate's definition.

That was a relatively easy step; a simple reformulation of each postulate in such a way as to reflect the new assumptions regarding the KB and the change was enough. This reformulation can be done in either syntactical terms (i.e., by reformulating the postulates syntactically) or in semantical terms (i.e., by expressing the postulates' underlying intuitions in a way that is applicable to our generalized model). Not surprisingly, both methods give the same results. The resulting postulates can be found in the following list, where the naming and numbering of each postulate corresponds to the original AGM naming and numbering:

(K−1)	$K-X=Cn(K-X)$	(*generalized closure*)
(K−2)	$K-X\subseteq Cn(K)$	(*generalized inclusion*)
(K−3)	If $X\notin Cn(K)$, then $K-X=Cn(K)$	(*generalized vacuity*)
(K−4)	If $X\notin Cn(\varnothing)$, then $X\notin Cn(K-X)$	(*generalized success*)
(K−5)	If $Cn(X)=Cn(Y)$, then $K-X=K-Y$	(*generalized preservation*)
(K−6)	$K\subseteq Cn((K-X)\cup X)$	(*generalized recovery*)

It can be trivially shown that, in the presence of the AGM assumptions, each postulate is equivalent to its generalized version. In fact, the original postulates differ from the generalized ones in two ways only: first, the original ones restrict the contracted expression to be a single proposition; second, the generalized ones use a slightly different notation to

accommodate their more general character. In the following, any reference to the AGM postulates will refer to their generalized version.

4.4. AGM-Compliance and Decomposability

AGM-Compliance

The introduction of the AGM postulates in [AGM85] was accompanied by the observation that every logic which satisfies the AGM assumptions can accommodate several contraction operators that satisfy the basic AGM postulates. Unfortunately, this result fails once the AGM assumptions are relaxed. In fact, in our wide framework, there are several logics in which no operator can be defined which satisfies all six (generalized) basic AGM postulates. This fact can be easily shown using the following example:

Example 4.1. Take a simple logic <L,Cn>, which allows only two propositions (L={a,b}), such that proposition b implies a but not vice-versa. More formally, the consequence operator is defined as:

$Cn(\emptyset)=\emptyset$

$Cn(\{a\})=\{a\}$

$Cn(\{b\})=\{a,b\}$

$Cn(\{a,b\})=\{a,b\}$

Using the inference relation $\vdash$, we could equivalently write:

$\{a,b\}\cong\{b\}\vdash\{a\}\vdash\emptyset$ and $\emptyset\nvdash\{a\}\nvdash\{b\}$.

It can be easily shown that the Cn operator defined above satisfies iteration, inclusion and monotony, thus, this <L,Cn> pair is a logic in the sense of Tarski. On the other hand, no operators (like $\neg$, $\vee$, $\wedge$, etc) can be defined in this logic, so it does not satisfy the AGM assumptions. Now take the sets of expressions K={a,b} and X={a} and set $K'=K-X$, for some operator $-$. Suppose that the operator '$-$' satisfies the generalized AGM postulates, so it is a contraction operator in the AGM sense. If $a\in K'$ then $X\subseteq K'$ which contradicts (K–4); if $b\in K'$, then, by (K–1): $K'=Cn(K')\supseteq Cn(\{b\})=\{a,b\}\supseteq X$, which again contradicts (K–4). Thus: $K'=Cn(K')=\emptyset$. But then, (K–6) is violated, since $Cn(K'\cup X)=Cn(\{a\})=\{a\}\subset K$. We conclude that no contraction operator can satisfy all six postulates at the same time, for these particular contraction operands in this particular logic.

Following this observation, our focus shifted on defining the conditions under which a logic admits a contraction operator satisfying all six postulates for all possible operands. The first step of this effort was to provide a special term for such logics, as well as for operators that satisfy the AGM postulates:

Definition 4.1. An operator $-:P(L)\times P(L)\rightarrow P(L)$ is an *AGM-compliant contraction operator with respect to the basic postulates for contraction* (or simply *AGM-compliant contraction operator*) iff it satisfies the basic AGM postulates for contraction (K–1)-(K–6).

A logic <L,Cn> is called *AGM-compliant with respect to the basic postulates for contraction* (or simply *AGM-compliant*) iff at least one AGM-compliant contraction operator can be defined in this logic.

Therefore, the major research question of this chapter is to formulate conditions under which a logic is (or is not) AGM-compliant. The following result shows that it is the debated postulate of recovery (K–6) that causes this problem:

Lemma 4.1. In any logic <L,Cn> there exists at least one operator that satisfies (K–1)-(K–5).

Proof

Consider the following function:

$K-X=Cn(K)$ iff $X \notin Cn(K)$

$K-X=Cn(\varnothing)$ iff $X \subseteq Cn(K)$

It is easy to show that this function can be defined in any logic and satisfies (K−1)-(K−5). □

Operators that satisfy (K−1)-(K−5) are called *withdrawal operators* [Mak87]. Thus, lemma 4.1 implies that a withdrawal operator can be defined in any logic. Notice however that the function used in the proof of the lemma is not really a "rational" contraction operator, because it causes a complete loss of previous knowledge whenever we try to contract something that is already in the KB (i.e., whenever $X \subseteq Cn(K)$). This policy is not accordant with the Principle of Minimal Change; not surprisingly, this operator does not satisfy the postulate of recovery.

Complement Beliefs and Decomposability

Once the recovery postulate is added to our list of desirable postulates, the result of lemma 4.1 is no longer true, as was shown by example 4.1. So, our next task is to find the conditions under which there is some function (operator) in the logic under question which satisfies all six postulates.

This can be accomplished through a closer examination of the postulates. Consider any logic <L,Cn> and two sets $K \subseteq L$ (the KB) and $X \subseteq L$ (the contracted belief). Set $K'=K-X$ and suppose that '−' is an AGM-compliant operator. Let us first consider some limiting cases:

- Suppose that $Cn(K)=Cn(\varnothing)$. By the inclusion postulate: $K' \subseteq Cn(\varnothing)$; by the closure postulate: $K'=Cn(K')$. Combining these facts we get that $K'=Cn(K')=Cn(\varnothing)=Cn(K)$.
- Suppose that $Cn(X)=Cn(\varnothing)$. By the recovery postulate: $K \subseteq Cn(K' \cup \varnothing)=Cn(K')$; by the inclusion postulate: $K' \subseteq Cn(K)$. Combining these two relations with the iteration and monotony properties of Cn we get that $Cn(K')=Cn(K)$, which, given the closure postulate, is equivalent to $K'=Cn(K)$.
- Suppose that $Cn(X) \notin Cn(K)$. It follows that $X \notin Cn(K)$, which, given the vacuity postulate, implies that $K'=Cn(K)$.

We conclude that in all three limiting cases, all we have to do is set $K'=Cn(K)$ to get a result that satisfies the AGM postulates. But things are not so simple in the more interesting case (and principal case for contraction), which occurs when $Cn(\varnothing) \subset Cn(X) \subseteq Cn(K)$. In this case, the postulates imply four main restrictions: first, the result K' should be closed under logical consequence (closure postulate); second, the result K' should be a subset of $Cn(K)$ (inclusion postulate); third, the closure of the result $Cn(K')$ should not contain X (success postulate); fourth, the result K' should be such that, after adding back X to it and closing under logical consequence, we should get some superset of K (recovery postulate).

Let us examine these conditions more closely. The inclusion postulate requires that $K' \subseteq Cn(K)$. If $K'=Cn(K)$, then $X \subseteq Cn(X) \subseteq Cn(K)=K'$, by our hypothesis and the inclusion postulate. This violates (K−4), so we conclude that the restriction of the inclusion postulate can be reduced to the condition: $K' \subset Cn(K)$.

Furthermore, since $K'=Cn(K')$, by the closure postulate, the latter condition is equivalent to the condition:

$Cn(K') \subset Cn(K)$ (1)

It can be shown that the recovery postulate and condition (1) imply success. Indeed, suppose, for the sake of contradiction, that $X \subseteq Cn(K')$; then, recovery gives:
$K \subseteq Cn(K' \cup X) = Cn(K')$
This fact implies: $Cn(K) \subseteq Cn(K')$.
The latter condition clearly contradicts (1), so the success postulate is redundant.

The recovery postulate can also be expressed in a more adequate manner. By our original hypothesis again, we have that $Cn(X) \subseteq Cn(K)$; combining this with (1), we get:
$Cn(K' \cup X) \subseteq Cn(Cn(K) \cup Cn(K)) = Cn(K)$
The latter condition, combined with recovery ($K \subseteq Cn(K' \cup X)$) implies:
$Cn(K) = Cn(K' \cup X)$ (2)

Summing up the above observations, we can conclude that the result of an AGM-compliant contraction operator, in the last, principal, case of contraction should have the following properties:

1) $K' = Cn(K')$ (by closure)
2) $Cn(K') \subset Cn(K)$ (by inclusion)
3) $Cn(K) = Cn(K' \cup X)$ (by modified recovery)

These restrictions show that the result of the contraction operation (K') must, in a sense, "fill the gap" between K and X. In other words, it must be the case that K can be "decomposed", with respect to X, in two sets X and K', such that both sets contain less knowledge than (are properly implied by) K when taken separately, but they have the same "informational strength" as (are equivalent to) K when combined (see figure 4.1). So, the result $K' = K - X$ could be viewed as a kind of "complement" of X with respect to K.

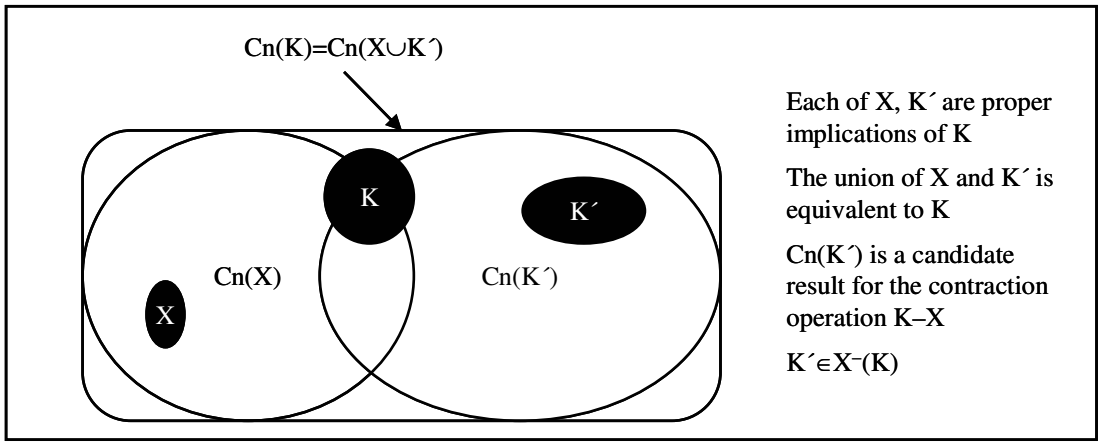

Figure 4.1. Intuition for decomposability

Our analysis shows that this kind of "complementarity" plays a central role for AGM-compliance: the result of an AGM-compliant contraction operation between two beliefs (sets) K, X should be a belief that is a "complement" of X with respect to K. We will prove this shortly, but, before that, we will need some definitions. The set of complement beliefs of a set with respect to another can be formally defined as follows:

Definition 4.2. Assume any logic <L,Cn> and two beliefs K, $X \subseteq L$. We define the set of *complement beliefs* of X with respect to K, denoted by $X^-(K)$ as follows:

- If $Cn(\varnothing) \subset Cn(X) \subseteq Cn(K)$, then $X^-(K) = \{Y \subseteq L \mid Cn(Y) \subset Cn(K) \text{ and } Cn(X \cup Y) = Cn(K)\}$
- In any other case, $X^-(K) = \{Y \subseteq L \mid Cn(Y) = Cn(K)\}$

Notice that the second bullet has been included for completeness and takes care of all the limiting cases. As the following lemma shows, $X^-(K)$ depends on the logical strength of X and K but not on their syntactical formulation:

Lemma 4.2. Consider a logic <L,Cn> and two beliefs K, $X \subseteq L$. Set $X_0=Cn(X)$, $K_0=Cn(K)$. Then:

- $X^-(K)=X_0^-(K_0)$
- $Y \in X^-(K)$ iff $Cn(Y) \in X^-(K)$

Proof

Obviously, $Cn(X)=Cn(X_0)$, $Cn(K)=Cn(K_0)$, $Cn(Y)=Cn(Cn(Y))$ and, by lemma 2.1, $Cn(X \cup Y)=Cn(X_0 \cup Y)$ for all $Y \subseteq L$. Using these facts, the results are obvious. □

Lemma 4.2 shows that equivalent beliefs give equal sets of complement beliefs; the second bullet justifies why the first requirement that resulted from our original analysis (i.e., that $K'=Cn(K')$) was omitted from the definition of complement beliefs. More properties of complement beliefs can be shown; for example:

Lemma 4.3. Consider a logic <L,Cn> and the beliefs K, X, $Y \subseteq L$. Then, the condition: $Cn(\varnothing) \subset Cn(X) \subseteq Cn(Y) \subseteq Cn(K)$ implies $X^-(K) \subseteq Y^-(K)$.

Proof

Consider some $Z \in X^-(K)$.

Then $Cn(Z) \subset Cn(K)$ and $Cn(Z \cup X)=Cn(K)$.

So: $Cn(K)=Cn(Z \cup X)=Cn(Z \cup Cn(X)) \subseteq Cn(Z \cup Cn(Y))=Cn(Z \cup Y) \subseteq Cn(Cn(K) \cup Cn(K))=Cn(K)$.

Thus $Z \in Y^-(K)$ and the proof is complete. □

The following lemma is a more general (but less straightforward) version of lemma 4.3:

Lemma 4.4. Consider a logic <L,Cn> and the beliefs K, X, $Y \subseteq L$, such that:

- $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$
- $Cn(\varnothing) \subset Cn(Y) \subset Cn(K)$
- There is no $Z \in X^-(K)$ such that $Cn(Z) \supseteq Cn(Y)$

Then $X^-(K) \subseteq Y^-(K)$.

Proof

Take a $Z \in X^-(K)$; then $Cn(Z) \subset Cn(K)$.

Suppose that $Z \notin Y^-(K)$. Set $Z'=Y \cup Z$.

Then:

$Cn(Z')=Cn(Y \cup Z) \subset Cn(K)$ (by the fact that $Z \notin Y^-(K)$)

$Cn(Z' \cup X)=Cn(Y \cup Z \cup X) \supseteq Cn(X \cup Z)=Cn(K)$

$Cn(Z' \cup X) \subseteq Cn(K \cup K)=Cn(K)$

Combining these relations we conclude that $Z' \in X^-(K)$.

But then $Z' \in X^-(K)$ and $Cn(Z')=Cn(Y \cup Z) \supseteq Cn(Y)$, which is a contradiction by the hypothesis.

We conclude that $Z \in Y^-(K)$, so $X^-(K) \subseteq Y^-(K)$. □

The following lemma shows that all beliefs implied by K and implying a complement belief $Y \in X^-(K)$ are also complement beliefs of X with respect to K:

Lemma 4.5. Consider a logic <L,Cn> and the beliefs K, X, $Y \subseteq L$ such that $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$ and $Y \in X^-(K)$. Then, $Cn(Y) \subseteq Cn(Y') \subset Cn(K)$ implies $Y' \in X^-(K)$.

Proof

By the definition of Y' it holds that $Cn(Y') \subset Cn(K)$.

Also $Y \in X^-(K) \Rightarrow Cn(X \cup Y)=Cn(K)$.

But $Cn(K)=Cn(X\cup Y)\subseteq Cn(X\cup Y')$ because $Cn(Y)\subseteq Cn(Y')$.
On the other hand $Cn(X)\subseteq Cn(K)$ and $Cn(Y')\subseteq Cn(K)$, thus $Cn(X\cup Y')\subseteq Cn(K)$.
In conclusion $Cn(K)=Cn(X\cup Y')$, so $Y'\in X^{-}(K)$. □

Decomposability and AGM-Compliance

The distinctive property that does not allow the logic of example 4.1 to admit an AGM-compliant operator is the existence of a pair of sets (namely $K=\{a,b\}$ and $X=\{b\}$) with an empty set of complement beliefs ($X^{-}(K)=\emptyset$). For this reason, no AGM-compliant operator could handle this particular contraction ($K-X$). The situation presented is typical in all non-AGM-compliant logics. Thus, beliefs (and logics) with non-empty sets of complement beliefs are important for our purposes, so we give them a special name:

Definition 4.3. Assume any logic $<L,Cn>$ and any set $K\subseteq L$.

- The set $K\subseteq L$ is *decomposable* iff $X^{-}(K)\neq\emptyset$ for every $X\subseteq L$
- The logic $<L,Cn>$ is *decomposable* iff for every $K, X\subseteq L$, $X^{-}(K)\neq\emptyset$ (or, equivalently, iff all $K\subseteq L$ are decomposable)

One immediate corollary of this definition and lemma 4.2 is that any set is decomposable iff its logical closure is. Therefore, either all equivalent sets are decomposable or all are non-decomposable. Another interesting result is that the condition $X^{-}(K)=\emptyset$ need not be checked for every pair of beliefs, as shown by the following lemma:

Lemma 4.6. Consider a logic $<L,Cn>$ and two beliefs $K, X\subseteq L$. If $X^{-}(K)=\emptyset$ then:
$Cn(\emptyset)\subset Cn(X)\subset Cn(K)$.

Proof

Initially suppose that the condition $Cn(\emptyset)\subset Cn(X)\subseteq Cn(K)$ is not true; then, by the definition of $X^{-}(K)$ it holds that $K\in X^{-}(K)$, so $X^{-}(K)\neq\emptyset$ (a contradiction).
Thus: $Cn(\emptyset)\subset Cn(X)\subseteq Cn(K)$.
Now suppose that $Cn(\emptyset)\subset Cn(X)=Cn(K)$.
Then for $Y=\emptyset$ we have that $Cn(Y)\subset Cn(K)$ and $Cn(X\cup Y)=Cn(X)=Cn(K)$; thus $\emptyset\in X^{-}(K)$, so $X^{-}(K)\neq\emptyset$ (a contradiction again).
We conclude that $Cn(\emptyset)\subset Cn(X)\subset Cn(K)$. □

A corollary of this lemma is that the condition $X^{-}(K)=\emptyset$ need only be checked for $X, K\subseteq L$ such that $Cn(\emptyset)\subset Cn(X)\subset Cn(K)$. Not surprisingly, this is also the principal case in a contraction operation $K-X$: the case in which we have problems with the recovery postulate. This result will be heavily used in the following. One of its initial applications appears in the following lemma, which verifies that, if $X^{-}(K)=\emptyset$, then all propositions in K (and in X) are implied by the propositions that are in K but not in X:

Lemma 4.7. Consider a logic $<L,Cn>$ and two beliefs $K, X\subseteq L$. If $X^{-}(K)=\emptyset$ then:
$Cn(K\backslash X)=Cn(K)$.

Proof

Since $X^{-}(K)=\emptyset$, we get, by lemma 4.6, that $Cn(\emptyset)\subset Cn(X)\subset Cn(K)$.
Set $Y=K\backslash X$.
Obviously, $Cn(Y)\subseteq Cn(K)$; suppose that $Cn(Y)\subset Cn(K)$.
Then $Cn(X\cup Y)=Cn(X\cup(K\backslash X))=Cn(K)$.
Thus, $Y\in X^{-}(K)$, a contradiction. So: $Cn(Y)=Cn(K)$. □

Lemma 4.7 is interesting because it allows us to show that $X^-(K)\neq\emptyset$ by calculating $Cn(K\backslash X)$; indeed, if $Cn(K\backslash X)\neq Cn(K)$, then lemma 4.7 implies that $X^-(K)\neq\emptyset$ (by contraposition).

We now have the tools necessary to show that the concepts of definition 4.1 and definition 4.3 actually coincide:

Theorem 4.1. A logic <L,Cn> is AGM-compliant iff it is decomposable.

Proof

($\Rightarrow$) Assume that the logic is AGM-compliant; then there exists a contraction function '$-$' that satisfies the basic AGM postulates for contraction.

By lemma 4.6, it suffices to show that for any K, $X\subseteq L$ for which $Cn(\emptyset)\subset Cn(X)\subset Cn(K)$ it is the case that $X^-(K)\neq\emptyset$.

Suppose any K, $X\subseteq L$ such that $Cn(\emptyset)\subset Cn(X)\subset Cn(K)$ and let $K'=K-X$.

We will show that $K'\in X^-(K)$.

By the postulate of inclusion: $K'\subseteq Cn(K)$

Suppose that $K'=Cn(K)$; then $X\subseteq Cn(K)=K'$, so the success postulate is violated.

Thus: $K'\subset Cn(K)$, which, using the postulate of closure, implies:

$Cn(K')\subset Cn(K)$ (1)

Furthermore, $K'\subseteq Cn(K)$ (by (1)) and $X\subseteq Cn(K)$ (by the hypothesis). Thus:

$Cn(K'\cup X)\subseteq Cn(K)$.

On the other hand, the recovery postulate gives:

$K\subseteq Cn(K'\cup X)$ and by taking the closure in both sides and applying iteration:

$Cn(K)\subseteq Cn(K'\cup X)$

The two inclusions: $Cn(K'\cup X)\subseteq Cn(K)$, $Cn(K)\subseteq Cn(K'\cup X)$ give:

$Cn(K)=Cn(K'\cup X)$ (2)

The relations (1) and (2) imply $K'\in X^-(K)$, so we are done.

($\Leftarrow$) Now assume that the logic is decomposable. We define a selection function s that selects one subset of L from a collection of subsets of L. Being a selection function, it has the property that: $s(\mathcal{B})\in\mathcal{B}$ for every collection $\mathcal{B}$ of subsets of L. If $\mathcal{B}$ is empty ($\mathcal{B}=\emptyset$), then $s(\mathcal{B})$ is not defined.

We also define a function $-:P(L)\times P(L)\rightarrow P(L)$ such that:

$K-X=Cn(s(X^-(K)))$

In short, the function '$-$' calculates the logical closure of an arbitrarily selected set from $X^-(K)$. The function '$-$' is well-defined because $X^-(K)\neq\emptyset$, by the hypothesis that the logic is decomposable.

Moreover, by lemma 4.2, it follows that $K-X\in X_0^-(K_0)$, where $X_0=Cn(X)$, $K_0=Cn(K)$, for all K, $X\subseteq L$.

Now, we can prove that the function '$-$' as defined is actually an AGM-compliant contraction function.

By the definition of '$-$', the postulate of closure (K$-$1) is obviously satisfied.

Since $K-X\in X^-(K)$, by the definition of $X^-(K)$ it follows that $K-X\subseteq Cn(K)$, so (K$-$2) is satisfied.

If $X\not\subseteq Cn(K)$, then $X^-(K)=\{Y\subseteq L \mid Cn(Y)=Cn(K)\}$, thus, the postulate of vacuity (K$-$3) holds.

For the postulate of success (K$-$4), we distinguish the different cases:

- Suppose that $Cn(\emptyset)\subset Cn(X)\subseteq Cn(K)$ and that $X\subseteq Cn(K-X)$. Since $Cn(K-X)\in X^-(K)$, we get: $Cn(K)=Cn(Cn(K-X)\cup X)=Cn(Cn(K-X))\subset Cn(K)$, a contradiction.

- In all other cases, since $K-X \in X^{-}(K)$, we get that $Cn(K-X)=Cn(K)$. Suppose, for the sake of contradiction, that $X \not\subseteq Cn(\varnothing)$ and $X \subseteq Cn(K-X)$. Then $X \subseteq Cn(K)$, thus $Cn(\varnothing) \subset Cn(X) \subseteq Cn(K)$, a contradiction by the hypothesis regarding X and K.

Thus, the postulate of success (K−4) holds too.

For the postulate of preservation (K−5), take two $X,Y \subseteq L$ such that $Cn(X)=Cn(Y)$. Then, by lemma 4.2, it follows that $X^{-}(K)=Y^{-}(K)$, so $K-X=K-Y$.

For the postulate of recovery (K−6), we will again distinguish the different cases:

- If $Cn(\varnothing) \subset Cn(X) \subseteq Cn(K)$, then, since $K-X \in X^{-}(K)$, by the definition of $X^{-}(K)$ we get that: $K \subseteq Cn(K)=Cn((K-X) \cup X)$.
- In any other case: $Cn((K-X) \cup X)=Cn(Cn(K) \cup X) \supseteq Cn(Cn(K)) \supseteq K$.

We conclude that '−' is an AGM-compliant contraction operator, so <L,Cn> is AGM-compliant. □

The proof of this theorem shows what we have already claimed to be true: that the candidate results for the operation K−X of an AGM-compliant contraction operator are *exactly* the logical closures of the sets in $X^{-}(K)$. Thus, if the set $X^{-}(K)$ is empty, no AGM-compliant contraction operation can handle K−X; if it is not empty, then any set in $X^{-}(K)$ can be chosen to produce an AGM-compliant result for K−X. This is more formally described in the following corollary:

Corollary 4.1. Consider a decomposable logic <L,Cn> and the operation $K-X=Cn(Y)$ for any K, $X \subseteq L$ and any arbitrarily selected $Y \in X^{-}(K)$. Then '−' is an AGM-compliant contraction operator.

Proof

The proof follows the same steps as the (⇐) direction of the proof of theorem 4.1 and is omitted. □

This corollary is very important because it shows that decomposability forms another representation result for the AGM postulates, like partial meet selection functions [AGM85], safe contraction [AM85], epistemic entrenchment [GM88], system of spheres [Gro88] and the postulates of Katsuno and Mendelzon [KM90b]. The distinctive property of decomposability as opposed to the other representation results is the fact that it is applicable to any logic, not just classical logics.

Another useful corollary results by combining theorem 4.1 with lemma 4.6:

Corollary 4.2. A logic <L,Cn> is AGM-compliant iff for all K, $X \subseteq L$ such that $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$, there is a $Y \subseteq L$ such that $Cn(Y) \subset Cn(K)$ and $Cn(X \cup Y)=Cn(K)$.

To put it simply, corollary 4.2 implies that each proper implication (say X) of any belief (say K) in an AGM-compliant logic can be "paired" with at least one other proper implication of K (say Y) which "complements" it, with respect to K; in this case, we will say that K is "decomposed" in X and Y (see also figure 4.1 in page 80 and figure 4.6 in page 96).

4.5. AGM-Compliance and Lattice Theory

AGM-Equivalent Logics

It is not rare in the literature to obtain results that show certain logics to be "equivalent"; such logics often have completely different syntax, have been developed using different terminology, are based on different formal frameworks and are being used in different

contexts or applications. One trivial such case occurs by applying a renaming scheme upon the namespace elements of a DL. More interesting such equivalences are described, for example, in [BCMNPS02], [Bor96] where certain DLs are shown equivalent with certain classical logics.

Equivalent logics in this sense can provably express exactly the same information, i.e., they are equally expressive. This is generally shown by applying a certain mapping between the propositions of each logic, such that the "meaning is preserved" during the mapping. To the author's knowledge, there is no standard definition of what formal properties such a mapping should have.

In this section, we will formally define a certain set of properties for such a mapping. Using this mapping, we can define a rather strong equivalence relation between logics, which fits both our intuition on "equivalence" and our needs in the following sections of this thesis. This definition of equivalence is applicable to all logics describable under Tarski's framework. Equivalent logics in our sense can express exactly the same information, so two equivalent logics have identical logical properties; not surprisingly, AGM-compliance is also preserved. The latter property of this equivalence relation is the important one for our purposes, so we will call such logics *AGM-equivalent*. The importance of this particular relation for this thesis is two-fold: first, AGM-equivalent logics have the same status as far as AGM-compliance is concerned; second, it allows us to show a certain interesting connection between AGM-compliance and lattice theory.

Obviously, the properties of a mapping that generates this equivalence relation should be based on both the set L and the consequence operator Cn of a logic $<L,Cn>$. To verify equivalence between two logics, we must find a mapping between the beliefs (as expressed by the set L) of these two logics with properties that guarantee that the structure (as expressed by the consequence operator Cn) is preserved during the mapping. This requirement can be formally expressed using the following definition:

Definition 4.4. Consider two logics $<L_1,\vdash_1>$, $<L_2,\vdash_2>$. The two logics will be called *AGM-equivalent* (or simply *equivalent*), denoted by $<L_1,\vdash_1>\sim<L_2,\vdash_2>$ iff there exists a mapping $f:P(L_1)\rightarrow P(L_2)$ with the following properties:

- For all $X, Y\subseteq L_1$, $X\vdash_1 Y \Leftrightarrow f(X)\vdash_2 f(Y)$
- For all $Z\subseteq L_2$, there exists a $Z_1\subseteq L_1$ such that $f(Z_1)\cong_2 Z$

If the mapping is relevant, we will write $<L_1,\vdash_1>\sim_f<L_2,\vdash_2>$, to denote that the two logics are *AGM-equivalent* via the mapping f.

In effect, for two logics $<L_1,Cn_1>$, $<L_2,Cn_2>$ to be equivalent, there must exist a mapping between the beliefs of these logics $f:P(L_1)\rightarrow P(L_2)$ with the following properties (see also figure 4.2):

- The ordering implied by the consequence operator (or, equivalently, the inference relation) of each logic should be preserved during the mapping.
- Each belief in $P(L_1)$ should be mapped to a belief in $P(L_2)$.
- Each belief in $P(L_2)$ should have a counterpart in $P(L_1)$, i.e., a belief in $P(L_1)$ which is mapped to it (or to one of its equivalent beliefs).

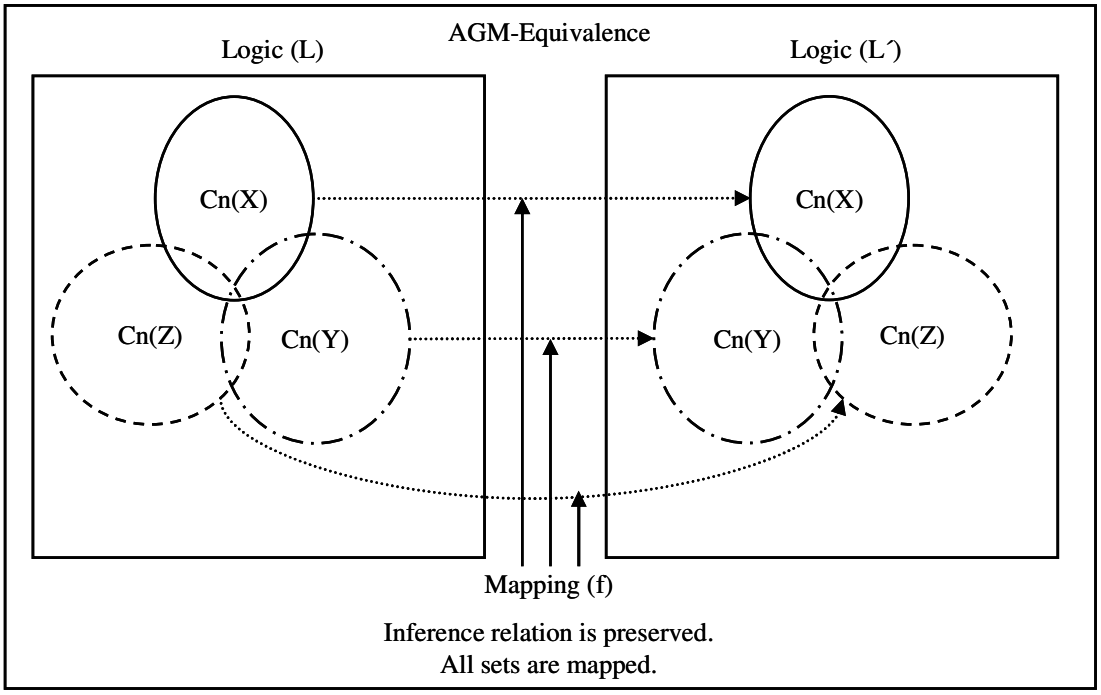

Figure 4.2. AGM-equivalence

It is trivial to see that the requirements of the definition are enough to preserve the whole structure of a logic during the mapping. The following lemma summarizes the relevant results; the proof is obvious using the definition of the equivalence relation and is omitted:

Lemma 4.8. If $<L_1,Cn_1>\sim_f<L_2,Cn_2>$, then for all $X, Y\subseteq L_1$ it holds that:

1) $X\vdash_1 Y \Leftrightarrow f(X)\vdash_2 f(Y)$
2) $X\cong_1 B \Leftrightarrow f(X)\cong_2 f(Y)$
3) $Cn_2(f(\varnothing))=Cn_2(\varnothing)$
4) $Cn_2(f(L_1))=Cn_2(L_2)=L_2$

AGM-Equivalence: Basic Properties

In most logics, there are several ways to express the same belief; in other words, there are several beliefs which are equivalent. Stripping a logic from such redundant sets and keeping one set per equivalence class, should result to an equivalent logic, because no expressive power has been removed. The following proposition shows that, in addition to this fact, the only way for two logics to be equivalent is for their equivalence classes to have the same cardinality and the same structure:

Proposition 4.1. Let $<L_1,Cn_1>$, $<L_2,Cn_2>$ two logics. Then $<L_1,Cn_1>\sim<L_2,Cn_2>$ iff there exists a $g:P(L_1)/\cong_1\rightarrow P(L_2)/\cong_2$ such that:

- g is 1-1 and onto
- For all $X, Y\subseteq L_1$ it holds that $[X]\vdash_1[Y] \Leftrightarrow g([X])\vdash_2 g([Y])$

Proof

($\Rightarrow$) Suppose that $<L_1,Cn_1>\sim_f<L_2,Cn_2>$.

Then, let $g:P(L_1)/\cong_1\rightarrow P(L_2)/\cong_2$ such that $g([X])=[f(X)]$ for all $[X]\in P(L_1)/\cong_1$.

Then for $X, Y\subseteq L_1$ we have that:

$g([X])=g([Y]) \Leftrightarrow [f(X)]=[f(Y)] \Leftrightarrow f(X)\cong_2 f(Y) \Leftrightarrow X\cong_1 Y \Leftrightarrow [X]=[Y]$.

Thus g is 1-1.

Let $Z_2\subseteq L_2$. Then, there exists a $Z_1\subseteq L_1$ such that:

$Cn_2(Z_2)=Cn_2(f(Z_1)) \Leftrightarrow Z_2\cong_2 f(Z_1) \Leftrightarrow [Z_2]=[f(Z_1)] \Leftrightarrow [Z_2]=g([Z_1])$.

Thus, for any $[Z_2]\in P(L_2)/\cong_2$ there is a $[Z_1]\in P(L_1)/\cong_1$ such that $g([Z_1])=[Z_2]$, so g is onto.
Finally, let $X, Y\subseteq L_1$. Then:
$[X]\vdash_1[Y] \Leftrightarrow X\vdash_1 Y \Leftrightarrow f(X)\vdash_2 f(Y) \Leftrightarrow [f(X)]\vdash_2[f(Y)] \Leftrightarrow g([X])\vdash_2 g([Y])$.
($\Leftarrow$) We define a function $f:P(L_1)\to P(L_2)$ with the property that for all $X\subseteq L_1$, $f(X)\cong_2 g([X])$. Obviously, such a function exists.
Then for $X, Y\subseteq L_1$ we get:
$X\vdash_1 Y \Leftrightarrow [X]\vdash_1[Y] \Leftrightarrow g([X])\vdash_2 g([Y]) \Leftrightarrow f(X)\vdash_2 f(Y)$.
Moreover, let $Z_2\subseteq L_2$. Since g is onto, there is a $[Z_1]\in P(L_1)/\cong_1$ such that:
$g([Z_1])=[Z_2] \Leftrightarrow f(Z_1)\cong_2 Z_2 \Leftrightarrow Cn_2(f(Z_1))=Cn_2(Z_2)$.
Thus $\langle L_1,Cn_1\rangle\sim_f\langle L_2,Cn_2\rangle$. □

Proposition 4.1 implies that the AGM-equivalence relation is a very strong one: two logics are equivalent if (and only if) the sets of their equivalence classes are isomorphic and they have the same "structure", as this is determined by the partial ordering implied by the inference relation (equivalently, the consequence operator). The strength of proposition 4.1 makes it quite difficult for two logics to be equivalent; despite that, several interesting equivalences do exist, as has been observed in the literature.

Furthermore, this proposition shows that our AGM-equivalence is actually not different from the standard equivalence relation commonly used in other contexts; it also coincides with our common intuition on what properties an "equivalence relation" should have. The contribution of this dissertation in this respect is the formal definition of this equivalence and the study of its properties with respect to AGM-compliance.

To justify its name, the relation ~ must be reflexive, symmetric and transitive. Indeed:
Proposition 4.2. The relation ~ is an equivalence relation, i.e., it is reflexive, symmetric and transitive.
Proof
For any logic $\langle L,Cn\rangle$ it holds that $\langle L,Cn\rangle\sim_f\langle L,Cn\rangle$ where f is the identity function: $f(X)=X$ for all $X\subseteq L$; thus ~ is reflexive.
Consider two logics $\langle L_1,Cn_1\rangle$, $\langle L_2,Cn_2\rangle$ such that: $\langle L_1,Cn_1\rangle\sim\langle L_2,Cn_2\rangle$. Then, by proposition 4.1 there is a function $g:P(L_1)/\cong_1\to P(L_2)/\cong_2$ that is 1-1, onto and for all $X, Y\subseteq L_1$ it holds that $[X]\vdash_1[Y]\Leftrightarrow g([X])\vdash_2 g([Y])$.
For the function $f=g^{-1}$, it holds that $f:P(L_2)/\cong_2\to P(L_1)/\cong_1$, f is 1-1, onto and for all $X, Y\subseteq L_2$ it holds that $[X]\vdash_2[Y] \Leftrightarrow g(g^{-1}([X]))\vdash_2 g(g^{-1}([Y])) \Leftrightarrow g^{-1}([X])\vdash_2 g^{-1}([Y]) \Leftrightarrow f([X])\vdash_2 f([Y])$.
Thus, by proposition 4.1 again $\langle L_2,Cn_2\rangle\sim\langle L_1,Cn_1\rangle$, so ~ is symmetric.
Finally, consider three logics $\langle L_1,Cn_1\rangle$, $\langle L_2,Cn_2\rangle$, $\langle L_3,Cn_3\rangle$ such that:
$\langle L_1,Cn_1\rangle\sim_f\langle L_2,Cn_2\rangle$, $\langle L_2,Cn_2\rangle\sim_g\langle L_3,Cn_3\rangle$.
Set $h:P(L_1)\to P(L_3)$, such that $h(X)=g(f(X))$ for all $X\subseteq L_1$. Then for all $X, Y\subseteq L_1$ we get:
$X\vdash_1 Y \Leftrightarrow f(X)\vdash_2 f(Y) \Leftrightarrow g(f(X))\vdash_3 g(f(Y)) \Leftrightarrow h(X)\vdash_3 h(Y)$.
Furthermore, take any $Z_3\subseteq L_3$. Then there is a $Z_2\subseteq L_2$ such that $g(Z_2)\cong_3 Z_3$.
For $Z_2\subseteq L_2$, there is a $Z_1\subseteq L_1$ such that $f(Z_1)\cong_2 Z_2$.
But: $f(Z_1)\cong_2 Z_2 \Leftrightarrow g(f(Z_1))\cong_3 g(Z_2) \Leftrightarrow h(Z_1)\cong_3 Z_3$.
So: $\langle L_1,Cn_1\rangle\sim_h\langle L_3,Cn_3\rangle$, thus ~ is transitive. □

One of the most important properties of the mapping that defines the equivalence relation is that even operations like $\cup$ or $\cap$ are preserved during the transition. This implies that two AGM-equivalent logics have the same structure. More specifically:

Proposition 4.3. Consider two logics $\langle L_1,Cn_1\rangle$, $\langle L_2,Cn_2\rangle$ such that $\langle L_1,Cn_1\rangle \sim_f \langle L_2,Cn_2\rangle$ and $X, Y \subseteq L_1$. Then:

- $f(X \cup Y) \cong_2 f(X) \cup f(Y)$
- $f(Cn_1(X) \cap Cn_1(Y)) \cong_2 Cn_2(f(X)) \cap Cn_2(f(Y))$

Proof

(For $\cup$) It holds that:

$X \cup Y \vdash_1 X$ and $X \cup Y \vdash_1 Y$, thus $f(X \cup Y) \vdash_2 f(X)$ and $f(X \cup Y) \vdash_2 f(Y)$.

Consider a $Z_2 \subseteq L_2$ such that $Z_2 \vdash_2 f(X)$, $Z_2 \vdash_2 f(Y)$.

Then there is a $Z_1 \subseteq L_1$ such that $f(Z_1) \cong_2 Z_2$.

The latter relation implies that:

$f(Z_1) \vdash_2 f(X)$ and $f(Z_1) \vdash_2 f(Y)$ thus $Z_1 \vdash_1 X$ and $Z_1 \vdash_1 Y$, which implies that:

$Z_1 \vdash_1 X \cup Y \Leftrightarrow f(Z_1) \vdash_2 f(X \cup Y) \Leftrightarrow Z_2 \vdash_2 f(X \cup Y)$.

Thus, $f(X \cup Y) \vdash_2 f(X)$, $f(X \cup Y) \vdash_2 f(Y)$ and for all $Z_2 \subseteq L_2$ such that $Z_2 \vdash_2 f(X)$, $Z_2 \vdash_2 f(Y)$ it holds that $Z_2 \vdash_2 f(X \cup Y)$.

Therefore: $f(X \cup Y) \cong_2 f(X) \cup f(Y)$.

(For $\cap$) It holds that:

$X \vdash_1 Cn_1(X) \cap Cn_1(Y)$ and $Y \vdash_1 Cn_1(X) \cap Cn_1(Y)$ thus:

$f(X) \vdash_2 f(Cn_1(X) \cap Cn_1(Y))$ and $f(Y) \vdash_2 f(Cn_1(X) \cap Cn_1(Y))$

Consider $Z_2 \subseteq L_2$ such that $f(X) \vdash_2 Z_2$, $f(Y) \vdash_2 Z_2$.

Then there is a $Z_1 \subseteq L_1$ such that $f(Z_1) \cong_2 Z_2$.

The latter relation implies that:

$f(X) \vdash_2 f(Z_1)$ and $f(Y) \vdash_2 f(Z_1)$ thus $X \vdash_1 Z_1$ and $Y \vdash_1 Z_1$, which implies that:

$Cn_1(X) \cap Cn_1(Y) \vdash_1 Z_1 \Leftrightarrow f(Cn_1(X) \cap Cn_1(Y)) \vdash_2 f(Z_1) \Leftrightarrow f(Cn_1(X) \cap Cn_1(Y)) \vdash_2 Z_2$.

Thus, $f(X) \vdash_2 f(Cn_1(X) \cap Cn_1(Y))$, $f(Y) \vdash_2 f(Cn_1(X) \cap Cn_1(Y))$ and for all $Z_2 \subseteq L_2$ such that $f(X) \vdash_2 Z_2$, $f(Y) \vdash_2 Z_2$ it holds that $f(Cn_1(X) \cap Cn_1(Y)) \vdash_2 Z_2$.

Therefore: $f(Cn_1(X) \cap Cn_1(Y)) \cong_2 Cn_2(f(X)) \cap Cn_2(f(Y))$. □

The following proposition provides a simple test to prove equivalence between two similar logics:

Proposition 4.4. Consider two logics $\langle L_1,Cn_1\rangle$, $\langle L_2,Cn_2\rangle$ such that:

- $L_1 \subseteq L_2$
- For all $X \subseteq L_1 \subseteq L_2$, it holds that $Cn_1(X) = Cn_2(X) \cap L_1$
- There is a function $f: L_2 \setminus L_1 \rightarrow P(L_1)$ such that for all $x \in L_2 \setminus L_1$ it holds that $Cn_2(\{x\}) = Cn_2(f(x))$

Then, $\langle L_1,Cn_1\rangle \sim \langle L_2,Cn_2\rangle$.

Proof

Set $f_0: L_2 \rightarrow P(L_1)$ such that:

$f_0(x) = \{x\}$, $x \in L_1$

$f_0(x) = f(x)$, $x \in L_2 \setminus L_1$

By the definition of f_0 and the hypothesis regarding function f, we conclude that for any $x \in L_2$ it holds that $Cn_2(\{x\}) = Cn_2(f_0(x))$.

Set $f_*: P(L_2) \rightarrow P(L_1)$ such that: $f_*(X) = \cup_{x \in X} f_0(x)$.

Then for any $X \subseteq L_2$, it holds that:

$Cn_2(X) = Cn_2(\cup_{x \in X}\{x\}) = Cn_2(\cup_{x \in X} Cn_2(\{x\})) = Cn_2(\cup_{x \in X} Cn_2(f_0(x))) =$

$= Cn_2(\cup_{x \in X} f_0(x)) = Cn_2(f_*(X))$.

Furthermore, it can be proven that for any $X, Y \subseteq L_1$ it holds that $X \vdash_1 Y \Leftrightarrow X \vdash_2 Y$.

Indeed, suppose that $X\vdash_1 Y$. Then $Cn_1(X)\subseteq Cn_1(Y)$. Take any $x\in Cn_2(X)$:
If $x\in L_1$, then $x\in Cn_2(X)\cap L_1=Cn_1(X)\subseteq Cn_1(Y)=Cn_2(Y)\cap L_1\subseteq Cn_2(Y)$. Thus: $x\in Cn_2(Y)$.
If $x\in L_2\backslash L_1$, then $f_0(x)\subseteq L_1$ and $Cn_2(\{x\})=Cn_2(f_0(x))$. Since $x\in Cn_2(X)$, we conclude that $f_0(x)\subseteq Cn_2(X)$. For any $y\in f_0(x)$, it holds that $y\in Cn_2(X)$ and $y\in L_1$, so by the previous case we conclude that $y\in Cn_2(Y)$, thus $f_0(x)\subseteq Cn_2(Y)$. This fact, along with the equality $Cn_2(\{x\})=Cn_2(f_0(x))$ implies that $x\in Cn_2(Y)$.
These two cases imply that $Cn_2(X)\subseteq Cn_2(Y)$, thus $X\vdash_2 Y$.
Now assume that $X\vdash_2 Y$. Then:
$Cn_2(X)\subseteq Cn_2(Y) \Rightarrow Cn_2(X)\cap L_1\subseteq Cn_2(Y)\cap L_1 \Rightarrow Cn_1(X)\subseteq Cn_1(Y) \Rightarrow X\vdash_1 Y$.
Thus $X\vdash_1 Y \Leftrightarrow X\vdash_2 Y$.
Set $g:P(L_1)\rightarrow P(L_2)$ such that $g(X)=X$ for all $X\subseteq L_1$.
Then: $X\vdash_1 Y \Leftrightarrow X\vdash_2 Y\Leftrightarrow g(X)\vdash_2 g(Y)$.
Take a $Z_2\subseteq L_2$. Then $f^*(Z_2)\subseteq L_1$ and $Cn_2(g(f^*(Z_2))=Cn_2(f^*(Z_2))=Cn_2(Z_2)$, so for any $Z_2\subseteq L_2$ there is a $Z_1=f^*(Z_2)\subseteq L_1$ such that $Cn_2(g(Z_1))=Cn_2(Z_2)$.
By the existence of g we conclude that $\langle L_1,Cn_1\rangle\sim\langle L_2,Cn_2\rangle$. □

The latter proposition implies the intuitively expected property that if we take a logic and add propositions which add no additional expressive power to it (i.e., they are "syntactic sugar"), then we get a logic which is AGM-equivalent to the original.

AGM-Equivalence and Decomposability

Our results form nice examples of useful properties that the AGM-equivalence relation has. However, our main concern is decomposability; for the AGM-equivalence relation to be useful for our purposes, it must preserve decomposability (and, consequently, AGM-compliance). This is true, as the following result shows:
Lemma 4.9. Consider two DLs $\langle L_1,Cn_1\rangle$, $\langle L_2,Cn_2\rangle$, such that $\langle L_1,Cn_1\rangle\sim_f\langle L_2,Cn_2\rangle$. If $X\subseteq L_1$ is decomposable, then $f(X)\subseteq L_2$ is decomposable.
Proof
If $Cn(X)=Cn(\varnothing)$ then X is (trivially) decomposable and $X\cong_1\varnothing \Leftrightarrow f(X)\cong_2 f(\varnothing) \Leftrightarrow Cn_2(f(X))=Cn_2(\varnothing)$, thus f(X) is (trivially) decomposable.
If $Cn(X)\neq Cn(\varnothing)$ then set $X'=f(X)$ and take any $Y'\subseteq L_2$ such that:
$Cn_2(\varnothing)\subset Cn_2(Y')\subset Cn_2(X')$.
If such a set does not exist, then X' is trivially decomposable and the proof is complete.
If it exists, then there is a $Y\subseteq L_1$ such that $f(Y)\cong_2 Y'$.
Then: $Cn_1(\varnothing)\subset Cn_1(Y)\subset Cn_1(X)$.
X is decomposable, so there is a $Z\subseteq L_1$ such that $Cn_1(Z)\subset Cn_1(X)$ and $Cn_1(Y\cup Z)=Cn_1(X)$.
Set $Z'=f(Z)\subseteq L_2$.
Then $Cn_1(Z)\subset Cn_1(X) \Leftrightarrow Cn_2(Z')\subset Cn_2(X')$ and
$Cn_1(Y\cup Z)=Cn_1(X) \Leftrightarrow$ (by definition)
$\Leftrightarrow Y\cup Z\cong_1 X \Leftrightarrow$ (by definition)
$\Leftrightarrow X'=f(X)\cong_2 f(Y\cup Z)\cong_2 f(Y)\cup f(Z)=Y'\cup Z' \Leftrightarrow$ (by definition and proposition 4.3)
$\Leftrightarrow Cn_2(Y'\cup Z')=Cn_2(X')$.
Thus, $Z'\in Y'^{-}(X') \Rightarrow Y'^{-}(X')\neq\varnothing$, so f(X) is decomposable. □

Lemma 4.9 has the following important corollary, which is actually the result we were seeking:

Corollary 4.3. Consider two logics $<L_1,Cn_1>$, $<L_2,Cn_2>$, such that: $<L_1,Cn_1>\sim_f<L_2,Cn_2>$. Then, the following are true:

- For all $X\subseteq L_1$ it holds that X: decomposable iff f(X): decomposable
- $<L_1,Cn_1>$ is decomposable iff $<L_2,Cn_2>$ is decomposable

Proof

The ($\Rightarrow$) direction of the first property follows from lemma 4.9; for the opposite, notice that $<L_2,Cn_2>\sim_g<L_1,Cn_1>$ for some function g (because ~ is symmetric), so if f(X) is decomposable then g(f(X)) is decomposable; since $g(f(X))\cong_1 X$, lemma 4.2 implies that X is decomposable.

The second property follows from the first. □

This result shows that the equivalence relation preserves decomposability (thus AGM-compliance). The mapping required for AGM-equivalence is so strong that either all equivalent logics are AGM-compliant or all are non-AGM-compliant. This will be very helpful in our subsequent analysis on examples of AGM-compliant and non-AGM-compliant logics, because once a logic has been proven to be AGM-compliant (or non-AGM-compliant) we can propagate this result to all its equivalent logics.

Lattice Theory

Another important property of the AGM-equivalence relation is that it makes possible to map every logic to a complete lattice. A lattice [Gra71] is a special type of *poset*. A poset is a set equipped with a *partial order* $\leq$, i.e., a relation satisfying *reflexivity* ($x\leq x$), *antisymmetry* ($x\leq y$ and $y\leq x$ implies $x=y$) and *transitivity* ($x\leq y$ and $y\leq z$ implies $x\leq z$). A *lattice* is a poset P with the additional property that any finite subset H of P ($H\subseteq P$) has a greatest lower bound (or *infimum*, denoted by infH) and a least upper bound (or *supremum*, denoted by supH). It can be proven [Gra71] that if infH and supH exist, then they are unique (see figure 4.3). A *complete lattice* is a poset P for which infH and supH exist for any set $H\subseteq P$ (not necessarily finite). We will denote lattices by the pair $<P,\leq>$. Two lattices $<P_1,\leq_1>$, $<P_2,\leq_2>$ are called *equivalent* (denoted by $<P_1,\leq_1>\sim<P_2,\leq_2>$) iff there exists a 1-1 and onto mapping $f:P_1\rightarrow P_2$ such that for any $x, y\in P_1$, $x\leq_1 y$ iff $f(x)\leq_2 f(y)$. For more detailed information on lattice theory see [Gra71].

The fact that any logic is equipped with a partial ordering relation (the inference relation: $\vdash$) implies a possible connection between logics and lattice theory. This connection has been studied in different contexts [BS81] and can be established by mapping each belief of the logic to an element in the lattice and using $\vdash$ (or its symmetric) as the partial order $\leq$ of the lattice. This way, equivalent beliefs will be mapped to the same element of the lattice, while the partial ordering $\vdash$ will be represented by the ordering $\leq$ of the lattice. In this subsection, we will formally establish this connection and prove that the concepts of lattice theory can be used in our study of AGM-compliance.

Take any logic $<L,\vdash>$ and the equivalence relation $\cong$ upon beliefs that is implied by $\vdash$. Set $P=P(L)/\cong$. We define the relation $\leq$ such that for any [X], [Y]$\in$P: $[X]\leq[Y]$ iff $[X]\vdash[Y]$. It can be shown that $<P,\leq>$ is a complete lattice. Now let $<P,\leq>$ be any complete lattice. We set L=P and for any X, $Y\subseteq L$ we set $X\vdash Y$ iff $\inf X\leq\inf Y$. It can be shown that the resulting structure $<L,\vdash>$ satisfies Tarski's axioms for being a logic.

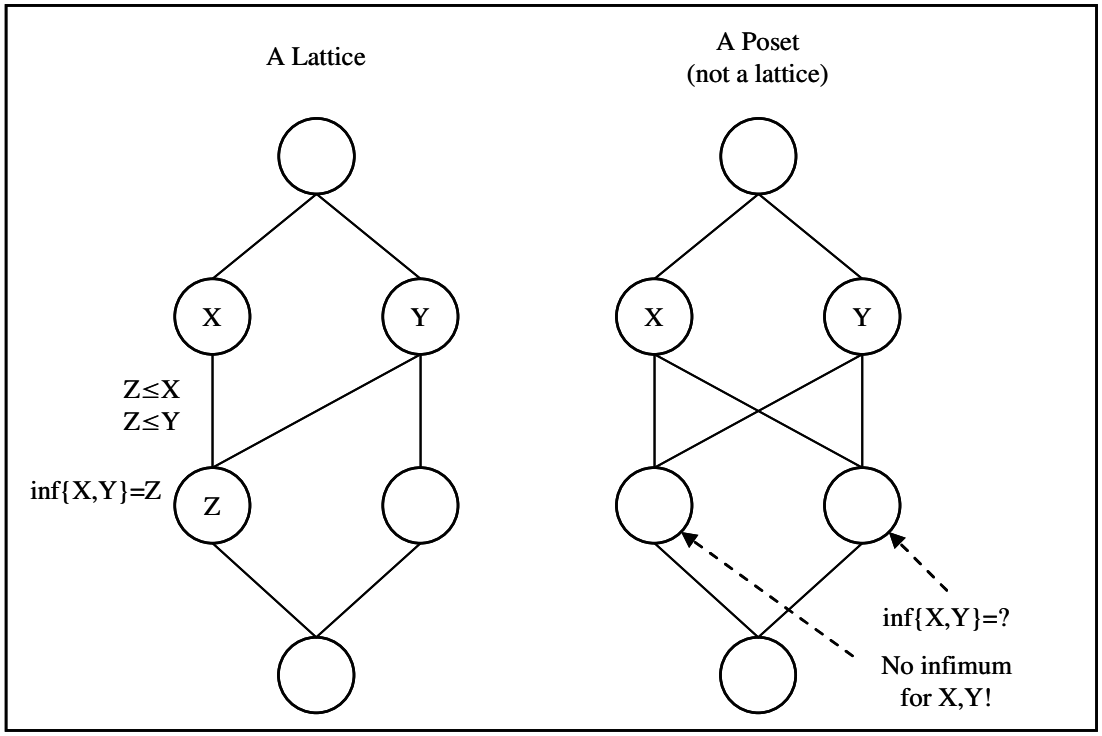

Figure 4.3. A lattice and a poset

The following lemmas use these two mappings to prove the connection between lattices and logics. We will use the following notation:

LAT_0={<P,≤> | <P,≤> is a complete lattice}, the set of complete lattices.
LAT=LAT_0/~, the quotient space of LAT_0 with respect to the equivalence relation on lattices (~).
LOG_0={<L,⊢> | <L,⊢> is a logic}, the set of logics that satisfy Tarski's axioms.
LOG=LOG_0/~, the quotient space of LOG_0 with respect to the AGM-equivalence relation on logics (~).

Using this terminology we can prove the following lemmas:

Lemma 4.10. There is a 1-1 mapping f:LOG→LAT.

Proof

We define: f:LOG→LAT such that for any class of logics [<L,⊢>]∈LOG:
f([<L,⊢>])=[<P,≤>], where P=P(L)/≅ and for any [X], [Y]∈P, [X]≤[Y] iff X⊢Y.
Initially, we must show that for every pair of logics $<L_1,\vdash_1>$, $<L_2,\vdash_2>$, such that:
$[<L_1,\vdash_1>]=[<L_2,\vdash_2>]$ it holds that $f([<L_1,\vdash_1>])=f([<L_2,\vdash_2>])$.
The two logics are equivalent, so, by proposition 4.1, there is a function $g:P(L_1)/\cong_1 \rightarrow P(L_2)/\cong_2$ with the properties described in proposition 4.1.
Equivalently, we can write g (and its properties) as follows:
$g:P_1 \rightarrow P_2$, g is 1-1 and onto and for all [X], [Y]∈P_1=$P(L_1)/\cong_1$ it holds that:
$[X]\leq_1[Y] \Leftrightarrow X\vdash_1 Y \Leftrightarrow [X]\vdash_1[Y] \Leftrightarrow g([X])\vdash_2 g([Y]) \Leftrightarrow g([X])\leq_2 g([Y])$
Thus $<P_1,\leq_1>\sim<P_2,\leq_2>$, so f is well-defined.
Secondly, we must prove that f([<L,Cn>])=[<P,≤>]∈LAT for any logic <L,Cn>∈LOG_0.
The relation ≤ is a partial order because:
For any [X]∈P, X⊢X, thus [X]≤[X], so reflexivity holds.
For any [X], [Y]∈P, if [X]≤[Y] and [Y]≤[X] then X⊢Y and Y⊢X, thus X≅Y ⇔ [X]=[Y], so antisymmetry holds.

For any [X], [Y], [Z]∈P, if [X]≤[Y] and [Y]≤[Z] then X⊢Y and Y⊢Z thus X⊢Z ⇔ X≤Z, so transitivity holds.
Therefore the pair <P,≤> is a poset.
Assume any set H⊆P. We set [Z]=[$\cup_{[X]\in H}$X].
By set theory, for any [X]∈H: Z⊢X ⇒ [Z]≤[X].
So, [Z] is a lower bound of H. Let any other lower bound [Z′].
Then, for any [X]∈H it holds that [Z′]≤[X] ⇒ Z′⊢X ⇒ Z′⊢$\cup_{[X]\in H}$X ⇒ [Z′]≤[$\cup_{[X]\in H}$X] ⇒
⇒ [Z′]≤[Z], so [Z] is a greatest lower bound of H, thus [Z]=infH.
Using lemma 14 in [Gra71] (page 29) we conclude that <P,≤> is a complete lattice.
Thus, <P,≤>∈LAT_0 ⇒ [<P,≤>]∈LAT.
Therefore, the range of f is as desired. It remains to show that f is 1-1.
Suppose that for [<L_1,Cn_1>], [<L_2,Cn_2>]∈LOG it holds that:
f([<L_1,Cn_1>])=f([<L_2,Cn_2>]).
Set [<P_1,$\leq_1$>]=f([<L_1,Cn_1>]), [<P_2,$\leq_2$>]=f([<L_2,Cn_2>]).
By f([<L_1,Cn_1>])=f([<L_2,Cn_2>]) we conclude that <P_1,$\leq_1$>~<P_2,$\leq_2$>.
So there is a mapping h:P_1→P_2 that is 1-1 and onto and has the property that for all [X], [Y]∈P_1=P(L_1)/$\cong_1$: [X]≤$_1$[Y] iff h([X])≤$_2$h([Y]).
Equivalently, we can write h (and its properties) as follows:
h:P(L_1)/$\cong_1$→P(L_2)/$\cong_2$, h is 1-1 and onto and for all [X], [Y]∈P(L_1)/$\cong_1$=P_1 it holds that:
[X]⊢$_1$[Y] ⇔ [X]≤$_1$[Y] ⇔ h([X])≤$_2$h([Y]) ⇔ h([X])⊢$_2$h([Y]).
Therefore, by proposition 4.1, <L_1,⊢$_1$>~<L_2,⊢$_2$>, so [<L_1,⊢$_1$>]=[<L_2,⊢$_2$>].
We conclude that f:LOG→LAT is 1-1. □

Lemma 4.11. There is a 1-1 mapping f:LAT→LOG.
Proof
We define f:LAT→LOG such that for any class of complete lattices [<P,≤>]∈LAT:
f([<P,≤>])=[<L,⊢>], where L=P and for any X, Y⊆L=P, X⊢Y iff infX≤infY.
<P,≤> is a complete lattice, so infX exists for all X⊆P.
Initially, we must show that for every pair of complete lattices <P_1,$\leq_1$>, <P_2,$\leq_2$>, such that:
[<P_1,$\leq_1$>]=[<P_2,$\leq_2$>] it holds that: f([<P_1,$\leq_1$>])=f([<P_2,$\leq_2$>]).
Since <P_1,$\leq_1$>~<P_2,$\leq_2$>, there is a function g:P_1→P_2 which is 1-1, onto and preserves the ordering (i.e., for all x, y∈P_1, x≤$_1$y iff g(x)≤$_2$g(y)).
Set [<L_1,⊢$_1$>]=f([P_1,$\leq_1$]), [<L_2,⊢$_2$>]=f([P_2,$\leq_2$]).
Define a function g′:P(L_1)/$\cong_1$→P(L_2)/$\cong_2$, as follows:
g′([X])=[{g(infX)}].
Then: [X]=$_1$[Y] ⇔ X$\cong_1$Y ⇔ infX=infY ⇔ g(infX)=g(infY) ⇔ g′([X])=g′([Y]), so g′ is 1-1.
Moreover, consider some [Z]∈P(L_2)/$\cong_2$ and set z=infZ∈P_2. There is some x∈P_1 such that g(x)=z. Set X={x}. Then g′([X])=[{g(infX)}]=[{g(x)}]=[{z}]=[Z] because infX=inf{x}=x. Thus, g′ is onto.
Finally, consider some [X], [Y]∈P(L_1)/$\cong_1$. Then:
[X]⊢$_1$[Y] ⇔ X⊢$_1$Y ⇔ infX≤$_1$infY ⇔ g(infX)≤$_2$g(infY) ⇔ g′([X])⊢$_2$g′([Y]).
Thus, by proposition 4.1, we conclude that <L_1,⊢$_1$>~<L_2,⊢$_2$>, thus: f([<P_1,$\leq_1$>])=f([<P_2,$\leq_2$>]).
Therefore, f is well-defined.
Secondly, we must show that f([<P,≤>])=[<L,⊢>]∈LOG for any complete lattice <P,≤>∈LAT_0.
To do that, we must prove that reflexivity, transitivity and weakening hold for ⊢.
Indeed:

Consider two X, Y⊆L such that X⊆Y; then infX≥infY, thus Y⊢X, so reflexivity holds.
Consider any X, Y, Z⊆L such that X⊢Y, X∪Y⊢Z. Then infX≤Y and inf(X∪Z)≤infZ. Consider any x∈X∪Z; if x∈X, then x≥infX; if x∈Y, then x≥infY≥infX. We conclude that inf(X∪Y)≥infX. But X⊆X∪Y, thus infX≥inf(X∪Y). Combining these relations: inf(X∪Z)=infX, thus infX≤infZ ⇔ X⊢Z, so transitivity holds.
Consider any X, Y, Z⊆L such that X⊢Y. Then infX≤infY. Since X⊆X∪Z we have that infX≥inf(X∪Z). Combining these results we get: inf(X∪Z)≤infY ⇔ X∪Y⊢Y, so weakening holds.
We conclude that $<L,Cn>\in LOG_0$, thus $[<L,Cn>]\in LOG$.
Therefore the range of f is as desired.
It remains to show that f is 1-1.
Indeed, suppose that for $[<P_1,\leq_1>], [<P_2,\leq_2>]\in LAT$ it holds that:
$f([<P_1,\leq_1>])=f([<P_2,\leq_2>])$.
Set $[<L_1,\vdash_1>]=f([<P_1,\leq_1>])$, $[<L_2,\vdash_2>]=f([<P_2,\leq_2>])$.
By $f([<P_1,\leq_1>])=f([<P_2,\leq_2>])$ we conclude that $<L_1,\vdash_1>\sim<L_2,\vdash_2>$.
So there is a mapping $g:P(L_1)\rightarrow P(L_2)$ such that for all X, $Y\subseteq L_1$: $X\vdash_1 Y$ iff $g(X)\vdash_2 g(Y)$ and for all $Z_2\subseteq L_2$ there exists a $Z_1\subseteq L_1$ such that $g(Z_1)\cong_2 Z_2$.
Set $h:P_1\rightarrow P_2$ such that $h(x)=inf_2g(\{x\})$ for all $x\in P_1=L_1$.
Consider any x, $y\in P_1$. If $h(x)=h(y) \Leftrightarrow inf_2g(\{x\})=inf_2g(\{y\}) \Leftrightarrow g(\{x\})\cong_2 g(\{y\}) \Leftrightarrow$
$\Leftrightarrow \{x\}\cong_1\{y\} \Leftrightarrow inf_1\{x\}=inf_1\{y\} \Leftrightarrow x=y$, so h is 1-1.
Consider any $y\in P_2$. Set $Z_2=\{y\}\subseteq L_2$. Then there exists a $Z_1\subseteq L_1$ such that $g(Z_1)\cong_2 Z_2$. Set $inf_1Z_1=x\in P_1=L_1$. Then $inf_1Z_1=inf_1\{x\} \Leftrightarrow Z_1\cong_1\{x\} \Leftrightarrow g(Z_1)\cong_2 g(\{x\})$.
Combining these relations, we conclude that:
$g(\{x\})\cong_2 Z_2 \Rightarrow inf_2g(\{x\})=inf_2Z_2=inf_2\{y\}=y \Rightarrow h(x)=y$, thus h is onto.
Finally take any x, $y\in P_1=L_1$. Then:
$x\leq_1 y \Leftrightarrow inf_1\{x\}\leq_1 inf_1\{y\} \Leftrightarrow \{x\}\vdash_1\{y\} \Leftrightarrow g(\{x\})\vdash_2 g(\{y\}) \Leftrightarrow$
$\Leftrightarrow inf_2g(\{x\})\leq_2 inf_2g(\{y\}) \Leftrightarrow h(x)\leq_2 h(y)$.
Therefore $<P_1,\leq_1>\sim<P_2,\leq_2>\Rightarrow[<P_1,\leq_1>]=[<P_2,\leq_2>]$.
We conclude that $f:LAT\rightarrow LOG$ is 1-1. □

The last two lemmas guarantee that there is a 1-1 mapping from LAT to LOG and vice-versa. This is enough to show the main result of this section:
Corollary 4.4. The spaces LOG and LAT are isomorphic. In other words, there is a 1-1 and onto mapping from LOG to LAT.
Proof
By lemma 4.10, there is a 1-1 mapping from LOG to LAT.
By lemma 4.11, there is a 1-1 mapping from LAT to LOG.
Combining these facts with the Schroder-Bernstein theorem, which can be found in any basic book on set theory (see [Hal74] or [End77] for example), we conclude that there is a 1-1 and onto mapping from LOG to LAT. □

Corollary 4.4, combined with corollary 4.3, imply that, as far as decomposability (and, consequently, AGM-compliance) is concerned, any logic can be equivalently represented as a complete lattice and vice-versa. In other words, AGM-compliance is a property exclusively related to the structure of the (complete) lattice that represents the logic, which is unique, modulo equivalence.

Indeed, take any logic that is, say, decomposable (thus AGM-compliant). Then all its equivalent logics are decomposable too and they are all mapped to the same (modulo equivalence) lattice; in effect, they are all mapped to the "same" lattice, because equivalent lattices have identical structure [Gra71]. The dual result holds if the logic is not AGM-compliant. So, instead of studying the properties of logics with respect to AGM-compliance, we could equivalently study the properties of lattices.

Using Lattices for Visualization

This alternative lattice representation of logics is very useful because it allows the use of the richness of results that have appeared in lattice theory [Gra71] to develop results on logics. The concepts and results of lattice theory that have been developed over the years can be used directly in our framework; this ability may allow the development of deeper results regarding AGM-compliant logics. Furthermore, using this mapping, we can reformulate all results presented in this thesis in terms of lattice theory; this reformulation is trivial and is omitted. Last, but certainly not least, lattices provide a nice and clear visualization to explain the intuition behind the theory of AGM-compliance. In the following, we will heavily use this representation for visualization purposes.

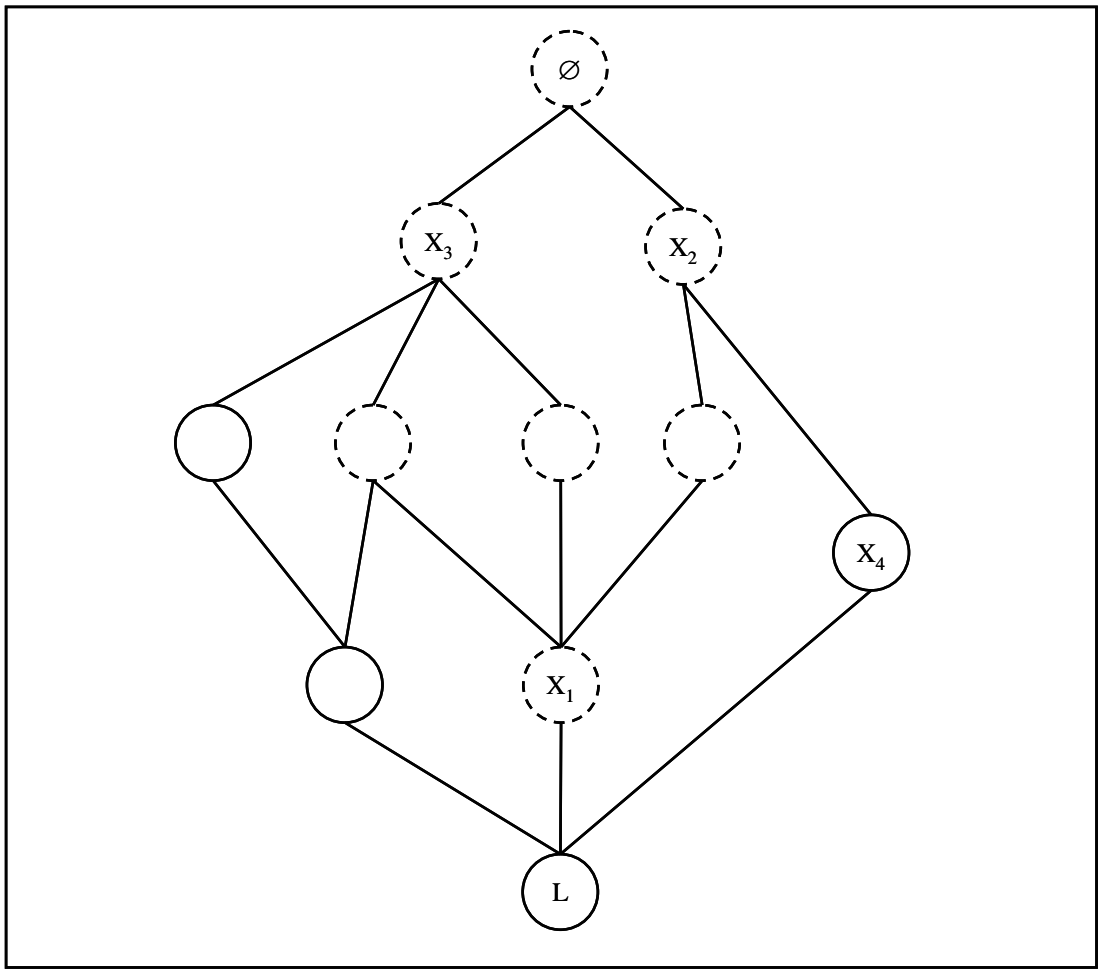

Figure 4.4. Lattices and logics

Figure 4.4 shows how a lattice can be used to visualize a logic. The top node of the lattice (1-element) corresponds to the tautologies, i.e., to the belief Cn(∅), or Cn(⊤). The bottom node (0-element) corresponds to the contradictions, i.e., the belief Cn(L), or Cn(⊥). The

implications of a node (belief) are all the nodes (beliefs) that are "above" it; in our example, the implications of X_1 are the dashed nodes. In lattice terminology these are the members of the set: $\{Y \mid X_1 \leq Y\}$; in logic terminology, it is the set: $\{Y \mid X_1 \vdash Y\}$. Similarly, $X_4 < X_2$, so $X_4 \vdash X_2$ and $X_2 \nvdash X_4$. The union of two beliefs (sets) is represented by the infimum in the lattice: the union of X_2 and X_3 is X_1 ($\inf\{X_2,X_3\}=X_1$), thus, $Cn(X_2 \cup X_3)=Cn(X_1)$. Similarly, the meet of two beliefs (sets) is the supremum: the meet of X_1 and X_4 is X_2 ($\sup\{X_1,X_4\}=X_2$), thus, $Cn(X_1) \cap Cn(X_4)=Cn(X_2)$. These notions can be extended for an arbitrary number of beliefs (even infinite number, since we are dealing with complete lattices).

Lattices can be used to represent well-known logics; figure 4.5 shows how PL, restricted to two atoms only, can be visualized in a lattice:

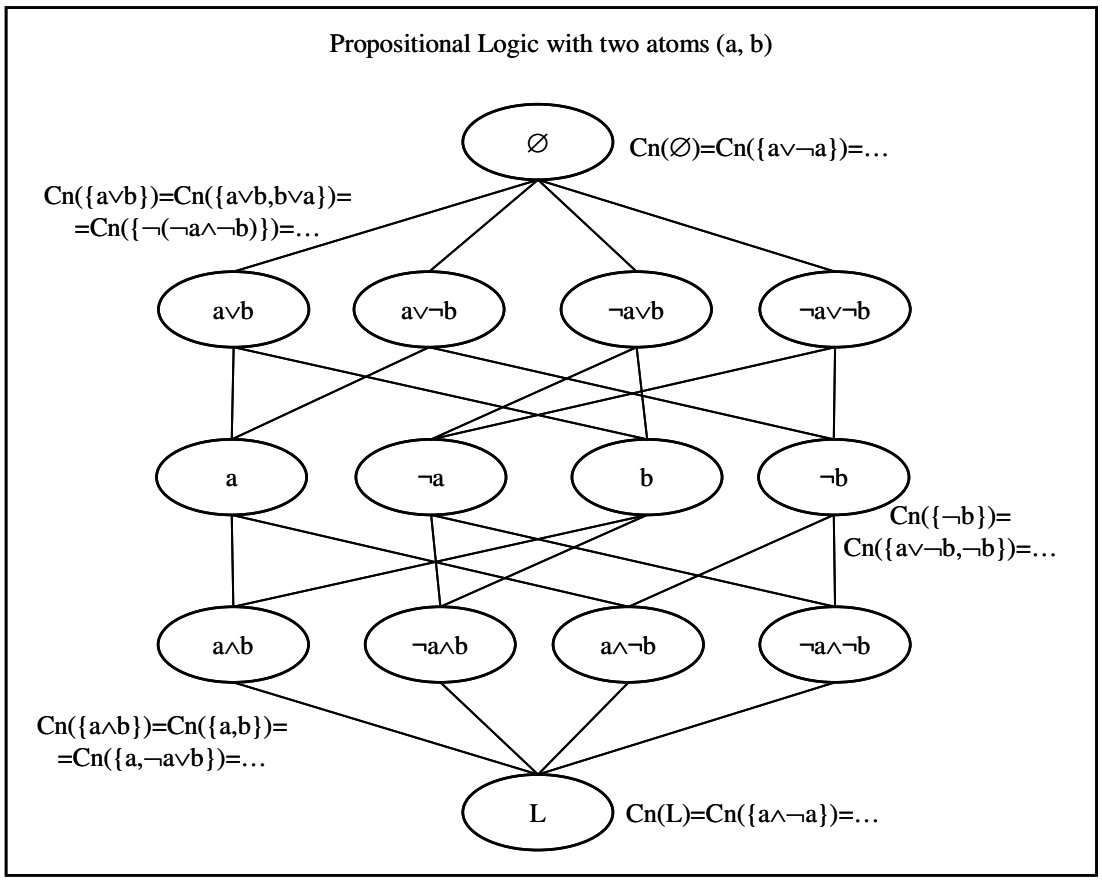

Figure 4.5. Visualizing Propositional Logic using a lattice

Figure 4.6 (see next page) shows how the notion of decomposability is represented in a lattice: a node that is in the complement set $X^-(K)$ should be "above" K in the lattice and the infimum of such a node with X should be K itself. Therefore, for example, X_5 complements X_1 (and X_1 complements X_5); X_3 is complemented by three different nodes, namely X_2, X_4 and X_5; while X_4 is complemented by X_3 and X_5. This visualization makes it easy to spot the nodes that belong to the set of complement beliefs of a given node (belief).

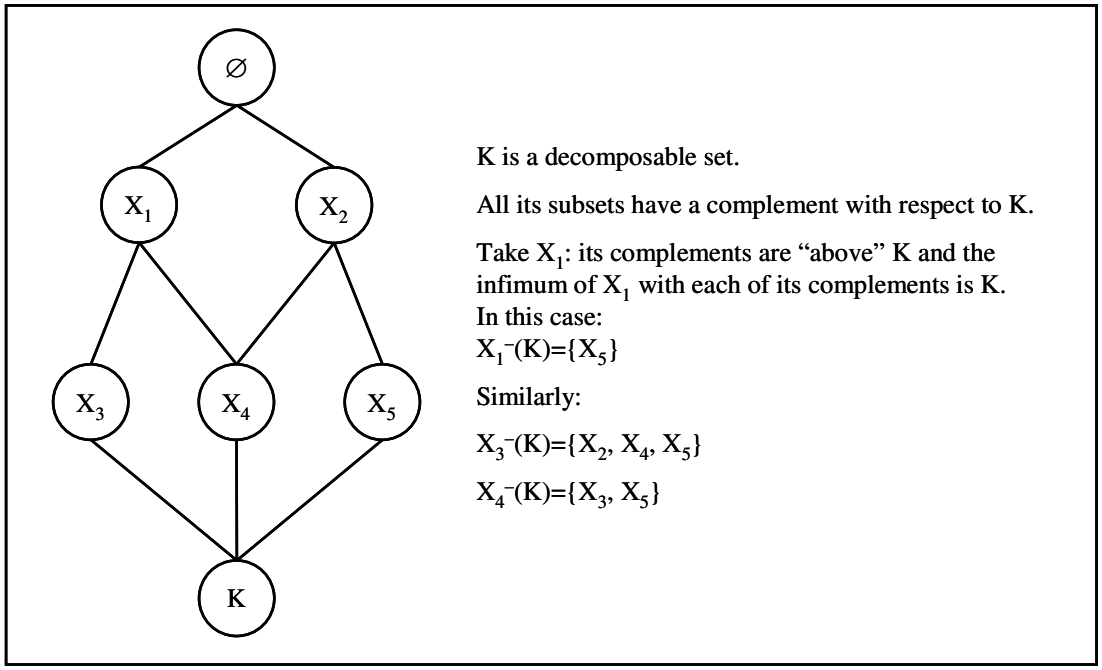

Figure 4.6. Decomposability in a lattice

The left part of figure 4.7 shows the representation of the (non-AGM-compliant) logic of example 4.1 in lattice form; notice that the belief {a} has no complement with respect to the belief L={a,b}. The right part shows that this logic can be rendered decomposable by the addition of an extra node in the lattice, which "decomposes" L={a,b} with respect to {a}:

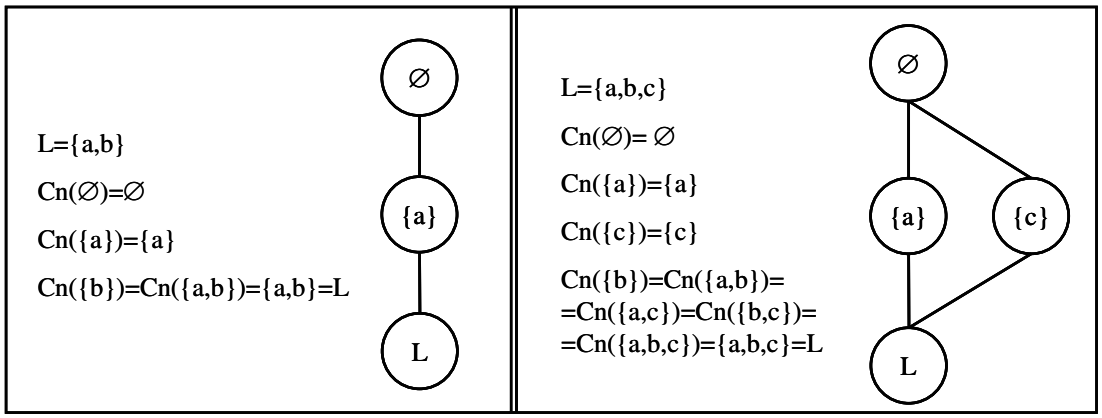

Figure 4.7. Visual representation of example 4.1

Figure 4.7 is exemplary on the intuition behind decomposability. Figure 4.4 is a more complicated example. In that figure, X_1 is decomposable; the only method we have to check this so far is by calculating $Y^-(X_1)$ for all Y such that $Cn(\varnothing)\subset Cn(Y)\subset Cn(X_1)$. For example, $X_3\in X_2^-(X_1)$, because $Cn(X_3)\subset Cn(X_1)$ and $Cn(X_3\cup X_2)=Cn(X_1)$. On the other hand, X_4 is not decomposable because $X_2^-(X_4)=\varnothing$; to see this, notice that the only nodes Y for which $Cn(Y)\subset Cn(X_4)$ (i.e., $Y>X_4$) are the nodes X_2 and $\varnothing$, for which $Cn(Y\cup X_2)=Cn(X_2)\neq Cn(X_4)$. Despite the fact that X_4 is not decomposable, its superset, L (i.e., the bottom node), is. Similarly, its subset, X_2, is trivially decomposable, because there is no $Y\subseteq L$ such that $Cn(\varnothing)\subset Cn(Y)\subset Cn(X_2)$. For similar reasons, X_3 is decomposable. Therefore, the supersets and subsets of X_4 are decomposable, while X_4 itself is not. Moreover, the node L is decomposable, while the logic itself <L,Cn> is not, because there is at least one belief in the

logic (namely X_4) which is not decomposable. Thus, we can't determine the AGM-compliance of a logic <L,Cn> by checking the decomposability of the set L itself; instead, we must check all the beliefs of the logic.

The following final example shows the intuitive meaning of the postulate (%2) that was introduced along with the existence assertion operator (%) and the CVA in section 3.3. As we can see, (%2) expresses the fact that all beliefs consisting entirely of existence assertion axioms are "above" the beliefs that contain at least one "traditional" DL axiom:

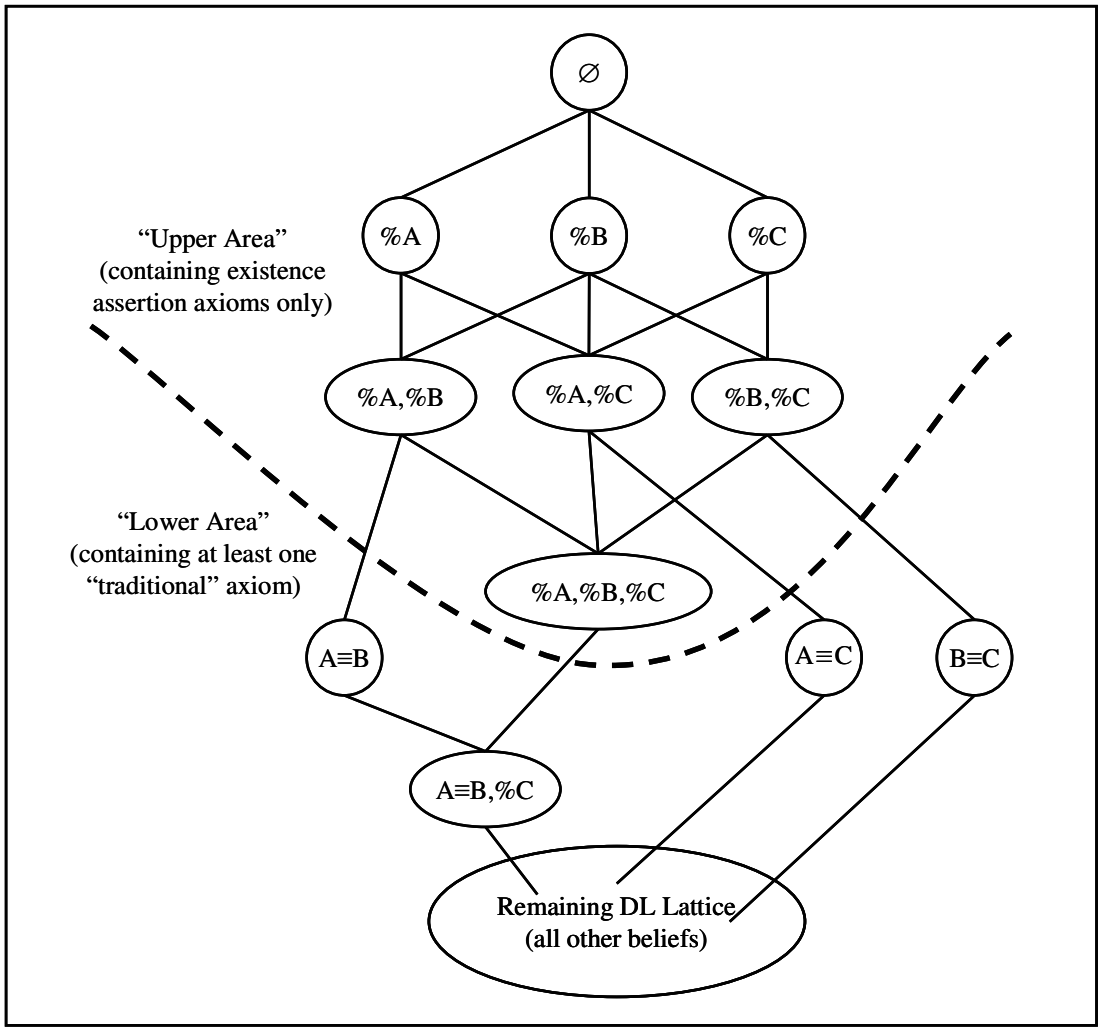

Figure 4.8. A typical DL under CVA

4.6. Alternative Characterizations: Cuts and Max-Cuts

Cuts

Apart from decomposability, two more alternative characterizations of AGM-compliance have been developed for the purposes of this thesis. The first one uses the concept of a "cut": a *cut* of a belief K is a family $\wp$ of beliefs that are properly implied by K with the property that any belief implied by K either implies or is implied by a member of the family $\wp$. More formally:

Definition 4.5. Consider any logic <L,Cn>, a belief $K \subseteq L$ and a family $\wp$ of beliefs ($\wp \subseteq P(L)$). $\wp$ will be called a *cut* of K iff:

- For all $Y \in \wp$, $Cn(Y) \subset Cn(K)$
- For all $Z \subseteq L$ such that $Cn(Z) \subset Cn(K)$ there is a $Y \in \wp$ such that $Cn(Z) \subseteq Cn(Y)$ or $Cn(Y) \subseteq Cn(Z)$

Cuts are important structures; figure 4.9 shows two possible cuts of a set K: one cut in the bottom of the lattice (double line), the other in the top (dashed line); their union is also a cut and many more cuts could be defined. $\wp$, for example, is a cut because any set Z implied by K ($Z \subseteq Cn(K)$) either implies ($Y \subseteq Cn(Z)$) or is implied ($Z \subseteq Cn(Y)$) by some set Y in $\wp$. Thus, a cut $\wp$, in a sense, "divides" the beliefs implied by A in two. The importance of cuts in our theory stems from their close connection with AGM-compliance.

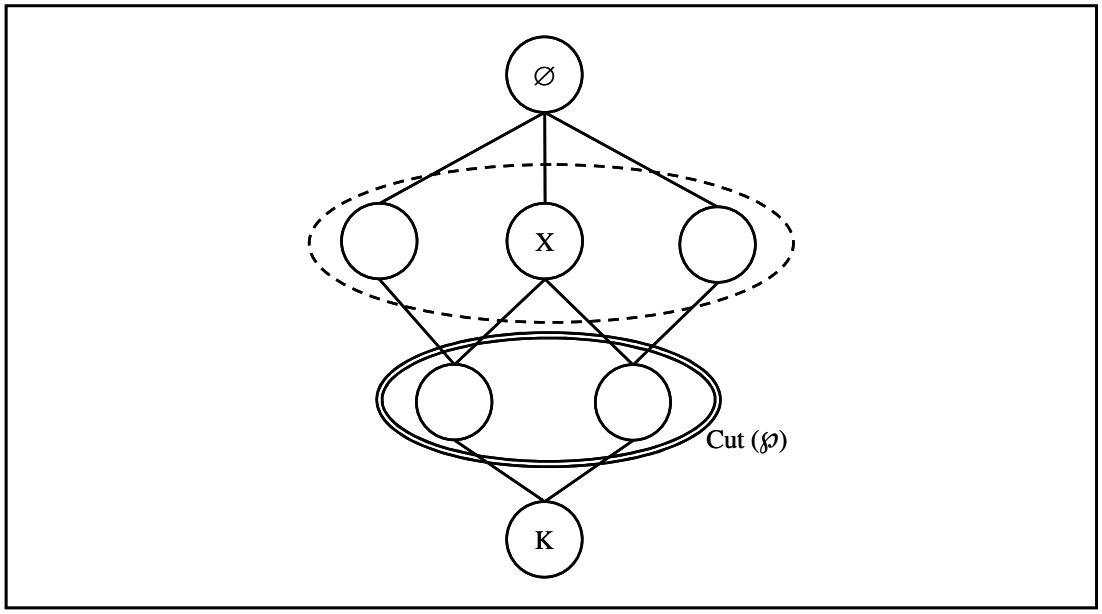

Figure 4.9. Examples of cuts

To see this connection, take any set $K \subseteq L$, a cut $\wp$ of K and a set X that is implied by all the sets in the cut $\wp$ (see also figure 4.9). Assuming that the operation $Z=K-X$ satisfies the AGM postulates, we get, by the inclusion postulate, that $Cn(Z) \subseteq Cn(K)$ (equivalently $K \vdash Z$). But then Z will either imply or be implied by a set $Y \in \wp$. If Z implies Y ($Z \vdash Y$), then $Z \vdash Y \vdash X$, so the result does not satisfy success (because $X \subseteq Cn(Z)$). If Z is implied by Y ($Y \vdash Z$), then $Y \vdash Z$, $Y \vdash X$, thus $Y \vdash X \cup Z$; but Y is a proper consequence of K, so recovery is not satisfied (because $Cn(X \cup Z) \subseteq Cn(Y) \subset Cn(K)$). Thus, the operation $K-X$ cannot be handled by any AGM-compliant contraction operator rendering the logic non-AGM-compliant. Once we deal with some technicalities and limit cases, it turns out that this is another equivalent characterization of AGM-compliance:

Theorem 4.2. Consider a logic <L,Cn> and a set $K \subseteq L$. Then K is decomposable iff for all cuts $\wp$ of K it is the case that $Cn(\cap_{Y \in \wp} Cn(Y)) = Cn(\varnothing)$.

Proof

($\Rightarrow$) Suppose that K is decomposable.

If $Cn(K) = Cn(\varnothing)$ then there is only one cut of K (namely $\wp = \varnothing$), so the theorem holds trivially. Suppose that $Cn(K) \neq Cn(\varnothing)$ and that for some cut $\wp$ of K it holds that $Cn(\cap_{Y \in \wp} Cn(Y)) = X \neq Cn(\varnothing)$. Obviously it holds that $X = Cn(X) \subseteq Cn(Y)$ for all $Y \in \wp$ thus $Cn(X) \subset Cn(K)$; therefore: $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$.

We will show that this set (X) has no complement with respect to K (i.e., $X^{-}(K) = \varnothing$).

Take any Z such that $Cn(Z) \subset Cn(K)$. By the definition of the cut, there exists a $Y \in \wp$ such that $Cn(Y) \subseteq Cn(Z)$ or $Cn(Z) \subseteq Cn(Y)$.
If $Cn(Y) \subseteq Cn(Z)$ then:
$Cn(X) \subseteq Cn(Y) \subseteq Cn(Z)$, so: $Cn(X \cup Z) = Cn(Z) \subset Cn(K) \Rightarrow Z \notin X^{-}(K)$.
If $Cn(Z) \subseteq Cn(Y)$ then:
$Cn(Z) \subseteq Cn(Y)$ and $Cn(X) \subseteq Cn(Y)$, so: $Cn(X \cup Z) \subseteq Cn(Cn(Y) \cup Cn(Y)) = Cn(Y) \subset Cn(K) \Rightarrow$
$\Rightarrow Z \notin X^{-}(K)$.
Thus, in any case, $Z \notin X^{-}(K)$, so $X^{-}(K) = \varnothing$, so K is not decomposable, a contradiction.
Thus, for any cut $\wp$: $Cn(\cap_{Y \in \wp} Cn(Y)) = Cn(\varnothing)$.
($\Leftarrow$) Now suppose that K is not decomposable. Then, by lemma 4.6, there exists a set $X \subseteq L$ such that $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$ and $X^{-}(K) = \varnothing$. We will construct a cut $\wp$ such that $Cn(\cap_{Y \in \wp} Cn(Y)) \neq Cn(\varnothing)$.
Take any $Z \subseteq L$ such that $Cn(Z) \subset Cn(K)$. Then $Cn(Z \cup X) \subseteq Cn(Cn(K) \cup Cn(K)) = Cn(K)$.
If $Cn(Z \cup X) = Cn(K)$, then, $Z \in X^{-}(K) \Rightarrow X^{-}(K) \neq \varnothing$, a contradiction.
So for all $Z \subseteq L$ such that $Cn(Z) \subset Cn(K)$ it holds that $Cn(Z \cup X) \subset Cn(K)$.
Take the family of beliefs: $\wp = \{Z \cup X \mid \text{for all } Z \subseteq L \text{ such that: } Cn(Z) \subset Cn(K)\}$.
For all $Y \in \wp$, $Cn(Y) \subset Cn(K)$, by the previous result.
Furthermore, for any $Z' \subseteq L$, $Cn(Z') \subset Cn(K)$ implies $Cn(Z') \subseteq Cn(Z' \cup X) \in \wp$.
Thus, $\wp$ is a cut. Furthermore, for all $Y \in \wp$: $Cn(Y) \supseteq Cn(X)$, therefore:
$Cn(\cap_{Y \in \wp} Cn(Y)) \supseteq Cn(X) \supset Cn(\varnothing)$.
So, we have constructed a cut with the desired property and the proof is complete. □

The following corollary is immediate:
Corollary 4.5. A logic <L,Cn> is AGM-compliant iff for all $K \subseteq L$ and all cuts $\wp$ of K it is the case that $Cn(\cap_{Y \in \wp} Cn(Y)) = Cn(\varnothing)$.

Max-cuts

Cuts have a very interesting and useful monotonic nature: if you replace any belief of a cut with some larger one(s) in forming a "larger" cut, then the quantity $Cn(\cap_{Y \in \wp} Cn(Y))$ cannot result to a smaller set (belief). Therefore, cuts with "larger" sets are more likely to have a non-tautological intersection of beliefs (compare for example the two cuts of figure 4.9). This observation motivates us to look for the "largest" possible cut. Indeed, consider a set K and its "largest possible" cut $\wp$: if $\wp$ has a non-tautological intersection, then K is non-decomposable (by theorem 4.2); if, on the other hand, $\wp$ has a tautological intersection, then all other, smaller, cuts will have a tautological intersection too (by monotonicity), so we can immediately infer that K is decomposable (by theorem 4.2) without having to check all possible cuts. To formalize this notion, we will need the definition of a *max-cut*:
Definition 4.6. Consider a logic <L,Cn>, a belief $K \subseteq L$ and a family $\wp$ of beliefs ($\wp \subseteq P(L)$) such that:

- For all $Y \in \wp$, $Cn(Y) \subset Cn(K)$
- For all $Z \subseteq L$ such that $Cn(Z) \subset Cn(K)$ there is a $Y \in \wp$ such that $Cn(Z) \subseteq Cn(Y)$
- For all $Y \in \wp$, $Cn(Y) = Y$
- For all X, $Y \in \wp$ $Cn(X) \subseteq Cn(Y)$ implies that $X = Y$

Then $\wp$ is called a *max-cut* of A.

Notice that the first two properties of a max-cut are the same as in the definition of a cut, except from the fact that the "or" clause in the second bullet has been dropped. Our intention

for a max-cut is to contain the maximal proper subsets of K, so there should be no implications of K which are supersets of a member of a max-cut. The latter two properties have been introduced for technical reasons: a max-cut contains exactly the maximal proper subsets of K, so it should be unique; however, this is only true modulo logical equivalence. To make things simpler, we constrain the sets of a max-cut to be theories (property #3) and to be non-comparable, in terms of the inference relation (property #4). In figure 4.9, the cut $\wp$ is also a max-cut. The properties that a max-cut has by definition, allow us to show that, if a max-cut exists, then it is unique:

Proposition 4.5. Consider a logic <L,Cn> and a belief $K \subseteq L$. If there exists a max-cut of K, then this max-cut is unique.

Proof

Suppose that there are two max-cuts of K, say $\wp_1$, $\wp_2$ and a $Y \in \wp_1$.

Consider some $Y \in \wp_1$.

Since $Y \in \wp_1$, we conclude that $Cn(Y) \subset Cn(K)$.

Since $\wp_2$ is a max-cut, there is a $Z \in \wp_2$ such that $Cn(Y) \subseteq Cn(Z)$.

Since $Z \in \wp_2$, we conclude that $Cn(Z) \subset Cn(K)$.

Since $\wp_1$ is a max-cut, there is a $Y' \in \wp_1$ such that $Cn(Z) \subseteq Cn(Y')$.

We conclude that Y, $Y' \in \wp_1$ and $Cn(Y) \subseteq Cn(Z) \subseteq Cn(Y')$, thus $Y=Y'$ and $Cn(Y)=Cn(Z)$.

Since $Y \in \wp_1$ and $\wp_1$ is a max-cut it follows that $Y=Cn(Y)$.

Since $Z \in \wp_2$ and $\wp_2$ is a max-cut it follows that $Z=Cn(Z)$.

Thus $Y=Cn(Y)=Cn(Z)=Z \in \wp_2$, which implies that for all $Y \in \wp_1$ we have that $Y \in \wp_2$.

Therefore $\wp_1 \subseteq \wp_2$.

Using similar arguments we get that $\wp_2 \subseteq \wp_1$, thus $\wp_1=\wp_2$, so the max-cut of K is unique. □

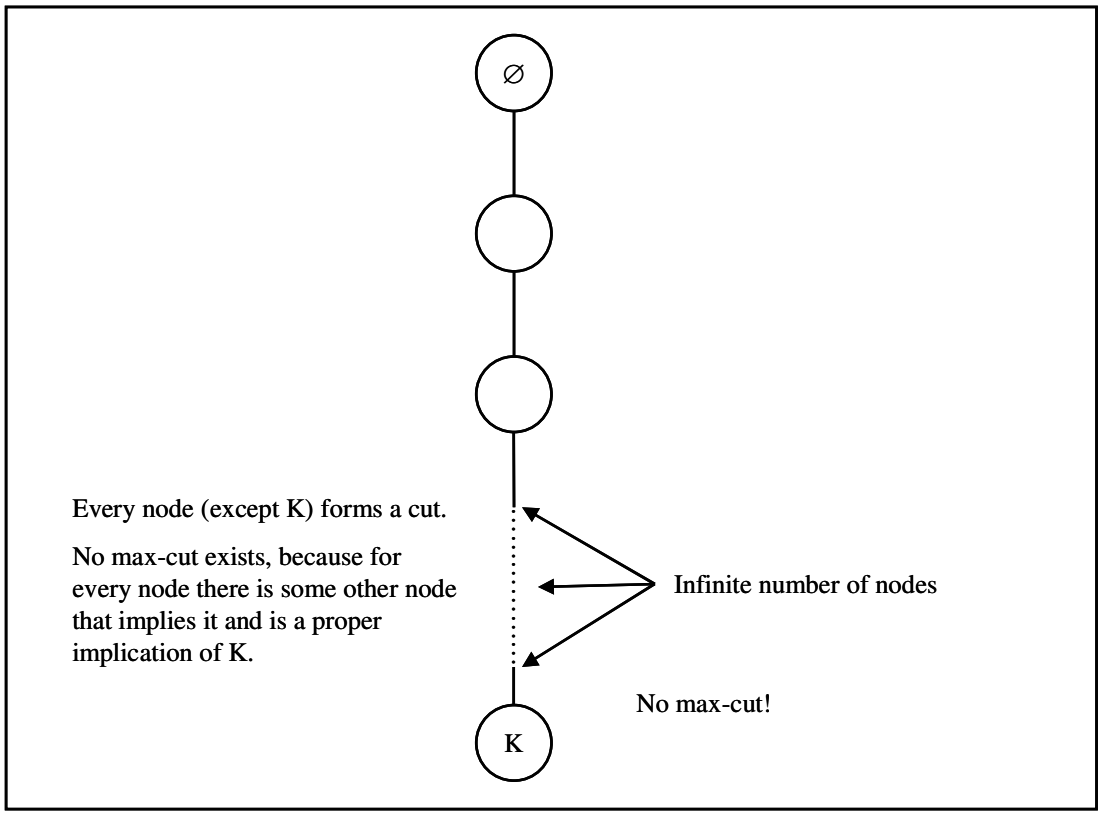

Figure 4.10. Max-cuts do not always exist

The "if exists" clause in the last proposition is very important: max-cuts contain all the maximal proper subsets of a set, which are not guaranteed to exist in logics with an infinite number of equivalence classes (see figure 4.10). Therefore, we cannot guarantee the existence of a max-cut in every set of every logic. A partial answer regarding the issue of the existence of a max-cut can be found in the following proposition:

Proposition 4.6. Consider any logic <L,Cn> and any belief $K \subseteq L$. If $P(Cn(K))/\cong$ is finite, then there exists a max-cut of K.

Proof

If $Cn(K)=Cn(\varnothing)$ then $\wp=\varnothing$ is a (trivial) max-cut, so there exists a max-cut.

If $Cn(K)\neq Cn(\varnothing)$, then there is at least one $Y \subseteq L$ such that $Cn(Y) \subset Cn(K)$ (for example set $Y=\varnothing$). Take any Y with this property.

We find all sequences of the form $Y_1, Y_2,\ldots,Y_n$ with the following properties:

- For all $i=1,2,\ldots,n$, $Y_i=Cn(Y_i)$
- For all $i=2,3,\ldots,n$, $Y_{i-1} \subset Y_i$
- $Y_1=Cn(Y)$ and $Y_n=Cn(K)$

In effect, we have found sequences of theories whose first member is equal to Cn(Y), its last member is equal to Cn(K) and every belief in every sequence is strictly larger than the previous one in the sequence.

There always exists such a sequence, for example (Cn(Y),Cn(K)) is a valid sequence.

Furthermore, all sequences have at least two members, namely Cn(Y) and Cn(K).

The fact that $P(Cn(K))/\cong$ is finite, guarantees that all such sequences will be finite and that there is a finite number of them.

Therefore we can select the sequence with the maximum number of elements (say m). If there is more than one with m elements we select one arbitrarily. Once again, the finiteness of $P(Cn(K))/\cong$ guarantees the existence of such a sequence.

We denote this sequence by $H_Y=(Y_1,Y_2,\ldots,Y_m)$ and set $Y_+=Y_{m-1}$.

We can immediately conclude that $Y_+=Cn(Y_+) \subset Cn(K)=Y_m$.

Furthermore $Cn(Y) \subseteq Y_+$ for all $Y \subseteq L$ such that $Cn(Y) \subset Cn(K)$.

Suppose that there exists some $Z \subseteq L$ such that $Cn(Y_+) \subset Cn(Z) \subset Cn(K)$.

Then the sequence $H'=(Y_1,Y_2,\ldots,Y_{m-1},Cn(Z),Y_m)$ would be a valid sequence with more elements than H_Y, a contradiction.

So, if there exists some $Z \subseteq L$ such that $Cn(Y_+) \subseteq Cn(Z) \subset Cn(K)$, then $Cn(Z)=Y_+$.

We set $\wp=\{Y_+ \mid Y \subseteq L, Cn(Y) \subset Cn(K)\}$.

We claim that $\wp$ is a max-cut.

Indeed, the following have been shown to be true:

- For all $Z \in \wp$, $Cn(Z) \subset Cn(K)$ (since $Cn(Y_+) \subset Cn(K)$ for any $Y \subseteq L$, $Cn(Y) \subset Cn(K)$)
- For all $Z' \subseteq L$ such that $Cn(Z') \subset Cn(K)$, there exists some $Z=Z'_+ \in \wp$ such that $Cn(Z') \subseteq Cn(Z'_+)=Cn(Z)$
- For all $Z \in \wp$, $Cn(Z)=Z$ (since $Cn(Y_+)=Y_+$ for all $Y \subseteq L$, $Cn(Y) \subset Cn(A)$)
- Take any Z, $Z' \in \wp$ and suppose that $Cn(Z) \subseteq Cn(Z')$. Suppose also that $Z=Y_+$, $Z'=Y'_+$ for some Y, $Y' \subseteq L$ with $Cn(Y) \subset Cn(K)$, $Cn(Y') \subset Cn(K)$. We conclude that $Z=Cn(Z) \subset Cn(K)$, $Z'=Cn(Z') \subset Cn(K)$. Since $Cn(Y_+)=Cn(Z) \subseteq Cn(Z') \subset Cn(K)$ we conclude (as shown before) that $Cn(Y_+)=Cn(Z') \Rightarrow Z=Z'$.

Thus $\wp$ is a max-cut and the proof is complete. □

Corollary 4.6. The following hold:

- Consider any logic <L,Cn> with the property that P(L)/≅ is finite and any belief K⊆L. Then, there exists a max-cut of K.
- Consider any logic <L,Cn> with the property that L/≅ is finite and any belief K⊆L. Then there exists a max-cut of K.
- Consider any logic <L,Cn> and any belief K⊆L. If Cn(K)/≅ is finite then there exists a max-cut of K.

The following theorem reveals the importance of max-cuts in our theory:

Theorem 4.3. Consider any logic <L,Cn> and a belief K⊆L. If there exists a max-cut $\wp$ of K, then K is decomposable iff $Cn(\cap_{Y\in\wp}Cn(Y))=Cn(\varnothing)$.

Proof

(⇒) It is obvious that a max-cut is also a cut. So, if K is decomposable then $Cn(\cap_{Y\in\wp}Cn(Y))=Cn(\varnothing)$ (by theorem 4.2).

(⇐) Now suppose that $Cn(\cap_{Y\in\wp}Cn(Y))=Cn(\varnothing)$.

Consider any other cut $\wp'$ of K.

For every $Y\in\wp$ it holds that $Cn(Y)\subset Cn(K)$, so there exists a $Y'\in\wp'$ such that $Cn(Y)\subseteq Cn(Y')$ or $Cn(Y')\subseteq Cn(Y)$.

Suppose that $Cn(Y)\subset Cn(Y')$. Since $Cn(Y')\subset Cn(K)$, there exists a $Z\in\wp$ such that $Cn(Z)\supseteq Cn(Y')\supset Cn(Y) \Rightarrow Cn(Z)\supseteq Cn(Y)$ and $Z, Y\in\wp$, which is a max-cut. We conclude that Y=Z, and $Cn(Y)\subset Cn(Y')\subseteq Cn(Z)$, an absurdity.

Thus, for every $Y\in\wp$ there exists a $Y'\in\wp'$ such that $Cn(Y')\subseteq Cn(Y)$.

Similarly, for every $Y'\in\wp'$ there exists a $Y\in\wp$ such that $Cn(Y')\subseteq Cn(Y)$, because $\wp$ is a max-cut.

Thus:

$Cn(\varnothing)=Cn(\cap_{Y\in\wp}Cn(Y))\supseteq Cn(\cap_{Y\in\wp'}Cn(Y))$, so:

$Cn(\cap_{Y\in\wp'}Cn(Y))=Cn(\varnothing)$ for all cuts $\wp'$ of K, thus K is decomposable (by theorem 4.2). □

Corollary 4.7. The following are true:

- Consider any logic <L,Cn> and a belief K⊆L with the property that P(Cn(K))/≅ is finite. Then K is decomposable iff for the max-cut $\wp$ of K it holds that: $Cn(\cap_{Y\in\wp}Cn(Y))=Cn(\varnothing)$.
- Consider any logic <L,Cn> such that P(L)/≅ is finite. The logic <L,Cn> is AGM-compliant iff for the max-cut $\wp$ of every K⊆L it holds that $Cn(\cap_{Y\in\wp}Cn(Y))=Cn(\varnothing)$.

Theorem 4.3, combined with the other result of this subsection, allows us to determine whether a set K⊆L is decomposable using only one test, namely by calculating $Cn(\cap_{Y\in\wp}Cn(Y))$ for the max-cut $\wp$ of K, provided that such a max-cut exists.

4.7. Discussion on AGM-Compliance

AGM Assumptions and AGM-Compliance

One of the crucial properties of the logics in the AGM class is that they all admit several contraction operators which satisfy the AGM postulates (both basic and supplementary). In the terminology of this thesis, this corresponds to the fact that all logics that satisfy the AGM assumptions should be AGM-compliant. In this subsection, we study the effects of the AGM assumptions on the issue of AGM-compliance, showing that even assumptions weaker than the AGM ones are enough to guarantee AGM-compliance; this implies the intuitively expected result that all logics that satisfy the AGM assumptions are AGM-compliant.

One of the key assumptions of AGM is that the usual operators ($\neg$, $\wedge$, $\vee$, $\rightarrow$, etc) can be defined in the underlying logic <L,Cn> with the usual semantics (classical tautological implication). Actually, the $\wedge$ operator can be expressed in Tarski's framework too, by forming the set that contains its operands: for example $\{a \wedge b\}$ is equivalent to $\{a, b\}$. The critical assumption missing in Tarski's framework is the existence of negation. Negation is also missing in most frameworks used for ontological representation (see [AADW05] for a relevant study and the introduction of negation for the RDF language).

Under the classical sense, the negation of a given proposition (say x) is a proposition ($\neg x$) which has the following properties:

- $Cn(\{x\}) \cap Cn(\{\neg x\}) = Cn(\varnothing)$
- $Cn(\{x\} \cup \{\neg x\}) = L$

Rephrasing these requirements in our more general terminology, we can define the negation of a belief as follows:

Definition 4.7. Consider a logic <L,Cn> and a set $K \subseteq L$. A set will be called a *negation* of K (denoted by $\neg K$) iff the following conditions hold:

- $Cn(K) \cap Cn(\neg K) = Cn(\varnothing)$
- $Cn(K \cup \neg K) = L$

Notice that in the general case that we consider, not all sets in a given logic have a negation, and, even if they have one, this is not necessarily unique (see figure 4.11). For this reason, the symbol $\neg$ should not be viewed as an operator, because an operator produces, by definition, a unique, well-defined result. In our context the symbol $\neg X$ is simply a denotation that follows the common scheme of classical logic.

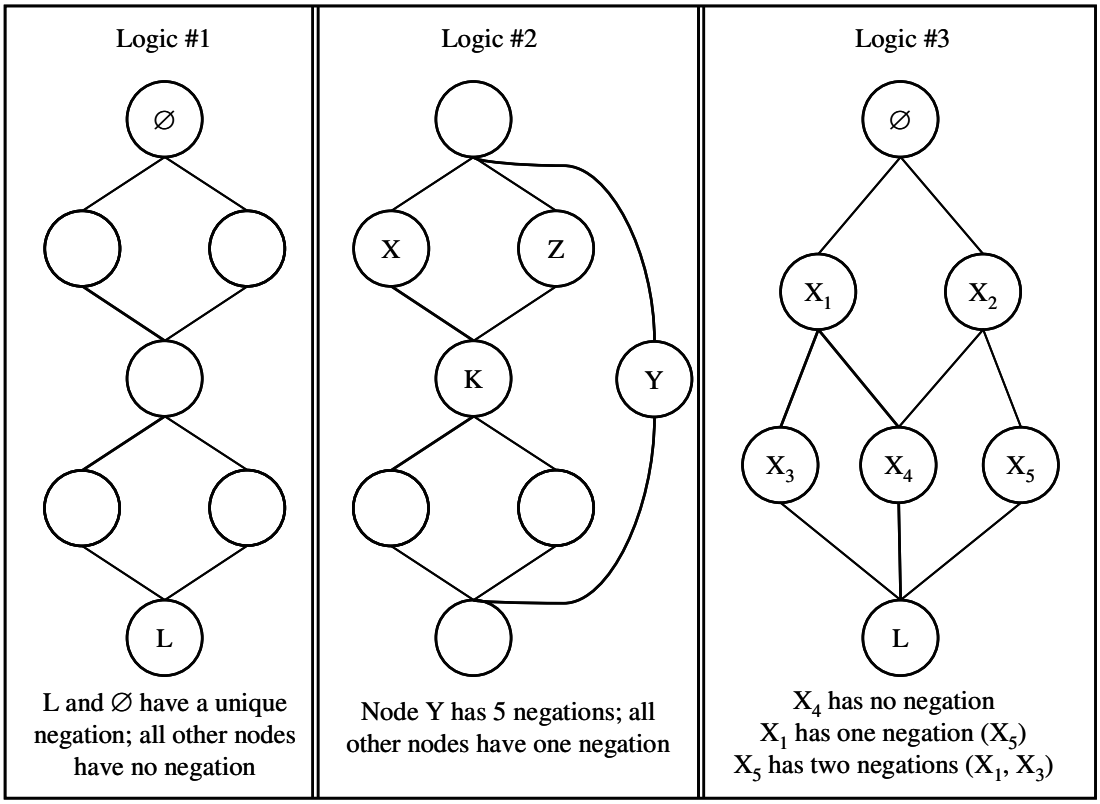

Figure 4.11. Negation in Tarski's framework

Negation is a very important concept: it is a well-known result from the classical logic literature that negation and conjunction (with the usual semantics), combined with the standard nice properties of classical logics eventually allow us to express all classical propositional operators ($\vee$, $\rightarrow$ etc) [Bur98]. For our purposes, negation is important because it allows us to find some set which, in a sense, contains "all beliefs that are not in X". Thus, the contraction of X from K (K−X) could be taken to be "everything that is in K, but not in X", or, equivalently, "everything that is in K and in $\neg$X"; the latter information can be expressed in Tarski's framework, by taking the belief $Cn(K)\cap Cn(\neg X)$ as the result of the contraction.

Our informal ideas can be formally shown only if the IDP rule (rule of introduction of disjunctions in the premises) is also assumed. In a different case, the structure of the underlying lattice could be so "exotic" that $\neg$X does not actually contain "all that is not in X" (this is true even if $\neg$X is unique). For example, in the second logic of figure 4.11, X has a unique negation (Y). Despite that, the belief that represents "everything that is in K and not in X" is the empty set ($\varnothing$), as $Cn(K)\cap Cn(Y)=Cn(\varnothing)$. Intuitively, we would expect this concept to be represented by Z, which is not the case, due to the exotic structure of the lattice.

On the other hand, the IDP rule forces a certain "nice" structure on the logic (lattice). The following initial result indicates this, by showing that the negation of a belief (if it exists) is unique (modulo logical equivalence), when the IDP rule holds:

Lemma 4.12. Consider a logic <L,Cn> in which the IDP rule holds and a set $X\subseteq L$. If there exist two negations of X, then these negations are equivalent.

Proof

Consider two negations of X, say X_1, $X_2\subseteq L$.

Then:

$Cn(X\cup X_1)=Cn(X\cup X_2)=L$ (1)

$Cn(X)\cap Cn(X_1)=Cn(X)\cap Cn(X_2)=Cn(\varnothing)$ (2)

It holds that:

$X_1\subseteq X_1\cup X_2\subseteq Cn(X_1\cup X_2)=Cn(Cn(X_1)\cup Cn(X_2))\cap L=$ (using (1))

$=Cn(Cn(X_1)\cup Cn(X_2))\cap Cn(Cn(X)\cup Cn(X_2))=$ (using IDP)

$=Cn(X_2\cup(Cn(X_1)\cap Cn(X)))=$ (using (2))

$=Cn(X_2\cup Cn(\varnothing))=Cn(X_2)$

Thus, $X_2\vdash X_1$.

Using similar arguments we can show that $X_1\vdash X_2$.

Thus $X_1\cong X_2$ and the proof is complete. □

This lemma implies that the introduction of IDP forces a certain structure on the logic, which, combined with the existence of a negation, is enough to guarantee AGM-compliance. This is shown in the following proposition:

Proposition 4.7. Consider any logic <L,Cn> in which the IDP rule holds and two sets K, $X\subseteq L$ such that X has a negation. Then $X^{-}(K)\neq\varnothing$.

Proof

As usual, the interesting case is when $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$.

Take one of the (equivalent) negations of X (i.e., $\neg$X) and set $Y=Cn(K)\cap Cn(\neg X)$.

Then $Cn(Y)\subseteq Cn(K)$.

Suppose that $Cn(Y)=Cn(K)$

Then:

$Cn(Y)\cap Cn(X)=Cn(K)\cap Cn(X)=Cn(X)$

But: $Cn(Y)\cap Cn(X)=Cn(K)\cap Cn(\neg X)\cap Cn(X)=Cn(K)\cap Cn(\varnothing)=Cn(\varnothing)$.
We conclude that: $Cn(X)=Cn(\varnothing)$, a contradiction.
Thus $Cn(Y)\subset Cn(K)$.
Furthermore:
$Cn(X\cup Y)=Cn(X\cup(Cn(K)\cap Cn(\neg X)))=$ (by IDP)
$=Cn(X\cup Cn(K))\cap Cn(X\cup Cn(\neg X))=Cn(Cn(K))\cap Cn(X\cup\neg X)=Cn(K)\cap L=Cn(K)$.
Thus $Y\in X^{-}(K) \Rightarrow X^{-}(K)\neq\varnothing$. □

The latter proposition has the following obvious corollary:
Corollary 4.8. Consider any logic <L,Cn> in which every belief has at least one negation and the IDP rule holds. Then <L,Cn> is AGM-compliant.

Notice that the assumptions of corollary 4.8 are weaker than the AGM assumptions, which eventually gives us the connection of AGM-compliance with the logics satisfying the AGM assumptions:
Corollary 4.9. All logics satisfying the AGM assumptions are AGM-compliant.

This corollary is important because it shows a property that was intuitively necessary for AGM-compliance, namely that the logics under the original AGM framework do satisfy the conditions for AGM-compliance.

Furthermore, it is important to notice that the above concepts have been studied in lattice theory [Gra71]: the negation of a belief in a logic corresponds to the complement of a node in a lattice; the IDP rule corresponds to the distributive identity in lattices and lattices that satisfy this identity are called distributive. Therefore, all logics that are represented by distributive, complemented, complete lattices (i.e., boolean lattices) are AGM-compliant. Notice that boolean lattices play an important role in lattice theory and their properties have been extensively studied (see [Gra71]).

Related Results and Discussion

In the previous sections we showed that AGM-compliance depends on the structure of the lattice that represents the logic at hand. There are three different equivalent ways to determine AGM-compliance; however, none of these methods allows us to determine AGM-compliance with one check for all the logic (lattice); each belief (node) of the logic (lattice) should be checked individually. In section 4.9 we will show two sufficient (but not necessary) conditions that allow this determination with one check.

Notice that the consequences of any belief (node), say K, form a (principal) dual ideal, denoted by ([K}], in the lattice that represents the logic (see [Gra71]). The structure of this principal dual ideal determines the decomposability of this particular belief (K); for a logic to be AGM-compliant, all its beliefs should form principal dual ideals (in the lattice) which have the required structure. If the belief K is decomposable, then one can define a contraction operator that handles the operation $K-X$ for all $X\subseteq L$ in an AGM-compliant manner; in order to extend this result for all $K\subseteq L$, all $K\subseteq L$ should be decomposable.

The different methods that we have provided for the determination of AGM-compliance have a different character and are most useful in different situations. Decomposability (theorem 4.1) provides a constructive method, which immediately leads to the definition of a contraction operator (see corollary 4.1); this means that decomposability is most useful when we are not simply interested in the existence of an operator, but want to find some example of

an AGM-compliant contraction operator as well. Cuts provide a more abstract method to show AGM-compliance (theorem 4.2). It is not easy to show AGM-compliance using cuts; cuts are most useful when we are trying to obtain a negative result. On the other hand, max-cuts allow us to obtain either a positive or a negative result with just one check (theorem 4.3); the downside is that max-cuts don't always exist, and, when they do, it is not always easy to find the max-cut and/or show that it actually is a max-cut. However, if we can easily find the max-cut of a belief, this method is superior to the other two. This is especially true if we suspect some belief to be decomposable, i.e., for obtaining positive results.

Our results (especially corollary 4.9) have shown that the class of AGM-compliant logics lies somewhere between the class we consider (i.e., Tarski's framework) and the AGM class (i.e., the class of logics that satisfy the AGM assumptions). Example 4.1 showed that the class of AGM-compliant logics does not coincide with the class of logics under Tarski's model; corollary 4.8 showed that it does not coincide with the class of logics that satisfy the AGM assumptions either. The following example is another manifestation of the latter fact:

Example 4.2. Consider some n>2 and a logic <L,Cn> such that:

$L=\{x_1, x_2, \ldots, x_n\}$

$Cn(\varnothing)=\varnothing$

$Cn(\{x_i\})=\{x_i\}$, for all i=1,2,…,n

$Cn(X)=L$ for all X with more than one element ($|X|\geq 2$)

It can be easily shown that Cn satisfies iteration, inclusion and monotony, so this structure is indeed a logic. Its lattice representation is shown in figure 4.12.

This logic is decomposable: the only case in which the condition $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$ holds is when X is a singular set and Cn(K)=L; but then $X^{-}(K)$ contains several sets, namely all singular sets except X itself. So the logic is decomposable.

On the other hand, <L,Cn> does not satisfy the AGM assumptions. One of the reasons is that each singular set has several negations (in effect, all other singular sets). This implies that the negation of a set cannot be uniquely defined (no negation operator $\neg$ can be defined) and that the IDP rule fails. This result can be easily generalized to the case where the set L is infinite (countable or uncountable).

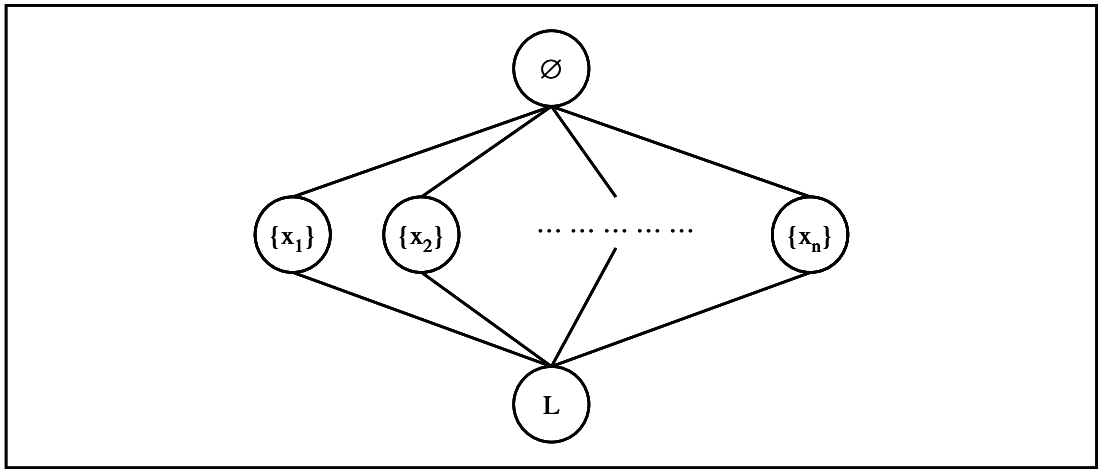

Figure 4.12. A decomposable logic not satisfying the AGM assumptions

Propagating AGM-Compliance

In certain cases, showing that a logic is AGM-compliant (or non-AGM-compliant) implies that a similar logic is also AGM-compliant (or non-AGM-compliant). This situation usually

appears when the two logics are AGM-equivalent (corollary 4.3). In this subsection, we will show certain conditions under which this propagation could occur without the logics being AGM-equivalent.

One such case appears when the beliefs of a logic <L,Cn> can be split in two groups (L_1, L_2), whose inference mechanisms (Cn_1, Cn_2) are largely independent:

Proposition 4.8. Consider two logics $<L_1,Cn_1>$, $<L_2,Cn_2>$ such that $L_1 \cap L_2 = \emptyset$. Define a third logic, <L,Cn> such that:

- $L = L_1 \cup L_2$
- For any $X \subseteq L$, $Cn(X) = Cn_1(X \cap L_1) \cup Cn_2(X \cap L_2)$

Then <L,Cn> is a logic and <L,Cn> is AGM-compliant iff both $<L_1,Cn_1>$ and $<L_2,Cn_2>$ are AGM-compliant.

Proof

Initially we must show that <L,Cn> is a logic.

Take any $X, Y \subseteq L$ and set $X_1 = X \cap L_1$, $X_2 = X \cap L_2$, $Y_1 = Y \cap L_1$, $Y_2 = Y \cap L_2$.

Then:

$X = X_1 \cup X_2 \subseteq Cn_1(X_1) \cup Cn_2(X_2) = Cn(X)$, so inclusion holds.

$Cn(Cn(X)) = Cn(Cn_1(X_1) \cup Cn_2(X_2)) =$

$= Cn_1((Cn_1(X_1) \cup Cn_2(X_2)) \cap L_1) \cup Cn_2((Cn_1(X_1) \cup Cn_2(X_2)) \cap L_2) =$

$= Cn_1(Cn_1(X_1)) \cup Cn_2(Cn_2(X_2)) = Cn_1(X_1) \cup Cn_2(X_2) = Cn(X)$, so iteration holds.

$X \subseteq Y \Rightarrow X_1 \subseteq Y_1$ and $X_2 \subseteq Y_2 \Rightarrow Cn_1(X_1) \subseteq Cn_1(Y_1)$ and $Cn_2(X_2) \subseteq Cn_2(Y_2) \Rightarrow$

$\Rightarrow Cn_1(X_1) \cup Cn_2(X_2) \subseteq Cn_1(Y_1) \cup Cn_2(Y_2) \Rightarrow Cn(X) \subseteq Cn(Y)$, so monotony holds.

Thus, <L,Cn> is a logic.

For the second result, suppose that <L,Cn> is AGM-compliant.

Take two sets K, $X \subseteq L_1$, such that $Cn_1(\emptyset) \subset Cn_1(X) \subset Cn_1(K)$.

Obviously $K, X \subseteq L$.

Also:

$Cn(\emptyset) = Cn_1(\emptyset) \cup Cn_2(\emptyset)$

$Cn(X) = Cn_1(X) \cup Cn_2(\emptyset)$

$Cn(K) = Cn_1(K) \cup Cn_2(\emptyset)$

Thus:

$Cn(\emptyset) \subset Cn(X) \subset Cn(K)$

Since <L,Cn> is AGM-compliant, there is a $Y \subseteq L$ such that:

$Cn(Y) \subset Cn(K)$ and $Cn(X \cup Y) = Cn(K)$

Set $Y_1 = Y \cap L_1$, $Y_2 = Y \cap L_2$.

Then $Cn(Y) = Cn_1(Y_1) \cup Cn_2(Y_2)$.

Since $Cn(Y) \subset Cn(K)$ it follows that:

$Cn_1(Y_1) \cup Cn_2(Y_2) \subset Cn_1(K) \cup Cn_2(\emptyset)$

The latter relation, combined with the facts that:

$Cn_1(Z) \subseteq L_1$ for all $Z \subseteq L_1$

$Cn_2(Z) \subseteq L_2$ for all $Z \subseteq L_2$

$L_1 \cap L_2 = \emptyset$

implies: $Cn_1(Y_1) \subset Cn_1(K_1)$.

Furthermore:

$Cn(K) = Cn(X \cup Y) = Cn_1(X \cup Y_1) \cup Cn_2(Y_2)$

Using the above relations again, we get:

$Cn_1(K) = Cn_1(X \cup Y_1)$.

These results imply that $Y_1 \in X^-(K)$ in $\langle L_1, Cn_1 \rangle$ so $\langle L_1, Cn_1 \rangle$ is AGM-compliant.
Using similar arguments, we can show that $\langle L_2, Cn_2 \rangle$ is AGM-compliant.
For the opposite result, suppose that $\langle L_1, Cn_1 \rangle$ and $\langle L_2, Cn_2 \rangle$ are AGM-compliant.
Take any two sets $K, X \subseteq L$ such that $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$.
Set $K_1 = K \cap L_1$, $K_2 = K \cap L_2$, $X_1 = X \cap L_1$, $X_2 = X \cap L_2$.
Take any $Y_1 \in X_1^-(K_1)$ in $\langle L_1, Cn_1 \rangle$ and any $Y_2 \in X_2^-(K_2)$ in $\langle L_2, Cn_2 \rangle$ and set $Y = Y_1 \cup Y_2$.
Then:
$Cn(Y) = Cn_1(Y_1) \cup Cn_2(Y_2) \subset Cn_1(K_1) \cup Cn_2(K_2) = Cn(K)$ (since $L_1 \cap L_2 = \varnothing$)
$Cn(X \cup Y) = Cn_1(X_1 \cup Y_1) \cup Cn_2(X_2 \cup Y_2) = Cn_1(K_1) \cup Cn_2(K_2) = Cn(K)$
Thus $Y \in X^-(K)$ in $\langle L, Cn \rangle$, so $\langle L, Cn \rangle$ is AGM-compliant and the proof is complete. □

The latter proposition has the following interesting corollary:

Corollary 4.10. Consider two logics $\langle L_1, Cn_1 \rangle$, $\langle L_2, Cn_2 \rangle$ such that:

- $L_1 \subseteq L_2$
- For any $X \subseteq L_2$, $Cn_2(X) = Cn_1(X \cap L_1) \cup (X \setminus L_1)$

Then $\langle L_1, Cn_1 \rangle$ is AGM-compliant iff $\langle L_2, Cn_2 \rangle$ is AGM-compliant.

Proof

If $L_1 = L_2$, the result is obvious. Suppose that $L_1 \subset L_2$.
Consider a logic $\langle L_3, Cn_3 \rangle$ such that:

- $L_3 = L_2 \setminus L_1$
- $Cn_3(X) = X$ for all $X \subseteq L_3$

It is trivial to show that $\langle L_3, Cn_3 \rangle$ is a logic.
To show that it is AGM-compliant, take any $K, X \subseteq L_3$ such that $Cn_3(\varnothing) \subset Cn_3(X) \subset Cn_3(K)$ and set $Y = K \setminus X$. It is trivial to show that $Y \in X^-(K)$ in $\langle L_3, Cn_3 \rangle$, so $\langle L_3, Cn_3 \rangle$ is AGM-compliant.
Furthermore, the following are true:
$L_1 \cap L_3 = \varnothing$
$L_1 \cup L_3 = L_2$
For any $X \subseteq L_2$, the following is true:
$Cn_2(X) = Cn_1(X \cap L_1) \cup (X \setminus L_2) = Cn_1(X \cap L_1) \cup (X \cap L_3) = Cn_1(X \cap L_1) \cup Cn_3(X \cap L_3)$.
These facts make proposition 4.8 applicable for the pair of logics $\langle L_1, Cn_1 \rangle$, $\langle L_3, Cn_3 \rangle$ and the logic $\langle L_2, Cn_2 \rangle$. Since $\langle L_3, Cn_3 \rangle$ is AGM-compliant by definition, it follows that $\langle L_1, Cn_1 \rangle$ is AGM-compliant iff $\langle L_3, Cn_3 \rangle$ is AGM-compliant (by proposition 4.8) and we are done. □

A further corollary of this result is the following, which will be helpful in our subsequent study of OWL (section 5.5):

Corollary 4.11. Consider two logics $\langle L_1, Cn_1 \rangle$, $\langle L_2, Cn_2 \rangle$ such that:

- $L_1 \subseteq L_2$
- For all $X \subseteq L_1$ it holds that $Cn_1(X) = Cn_2(X)$
- For all $X \subseteq L_1$ and all $Y \subseteq L_2 \setminus L_1$ it holds that $Cn_2(X \cup Y) = Cn_2(X) \cup Y$
- For all $Z \subseteq L_2$ and all $Y \subseteq L_2 \setminus L_1$ it holds that $Y \subseteq Cn_2(Z)$ iff $Y \subseteq Z$

Then $\langle L_1, Cn_1 \rangle$ is AGM-compliant iff $\langle L_2, Cn_2 \rangle$ is AGM-compliant.

Proof

Take any $Z \subseteq L_2$.
If $Z \subseteq L_1$ then $Cn_2(Z) = Cn_1(Z) = Cn_1(Z \cap L_1) \cup (Z \setminus L_1)$.
In a different case, set $Z_1 = Z \cap L_1$, $Z_2 = Z \setminus Z_1$.
Then: $Cn_2(Z) = Cn_2(Z_1 \cup Z_2) = Cn_2(Z_1) \cup Z_2 = Cn_1(Z \cap L_1) \cup (Z \setminus L_1)$.
Combining these results with corollary 4.10 gives the result. □

4.8. AGM Theory and the Foundational Model

Belief Base Operations and the AGM Theory

One of the criticisms that the AGM model had to face was the fact that theories are infinite structures, so no efficient algorithm based entirely on the AGM model could be developed ([Han99b], [Han92], Han96]). Furthermore, some authors ([Fuhr91], [Neb89]) state that the coherence model is inappropriate for belief change, because our knowledge regarding a domain stems from a small, finite number of observations, facts, experiments, rules, measurements, laws etc regarding the given domain; the rest of our beliefs are simply derived from such facts and should be removed once their logical support is lost. Therefore, the foundational model is more appropriate for the representation of our knowledge than the coherence model adopted by AGM.

As already mentioned, the foundational model leads to different belief change operators, the *belief base operators*, which don't require that the KB is necessarily a theory. Under this model, KBs are belief bases (instead of belief sets) and there is an important discrimination between explicit and implicit knowledge. Explicit knowledge represents the facts that are explicitly stored in the KB; these are acquired directly from observations, facts, experiments, etc. Implicit knowledge is not stored in the KB and can be accessed only by calculating the logical consequences of the explicit knowledge. Most importantly, only explicit knowledge can be directly changed during any belief change operation; implicit knowledge can only be indirectly affected through changes in the explicit knowledge.

In particular, when contracting a belief base, only elements (propositions) from the base itself can be removed; once this removal is complete we can recalculate the consequences of the new base (the new implicit facts). Belief change operations on belief bases appeared as a reasonable alternative to the AGM model due to their nice computational properties and intuitive appeal.

Base-AGM-Compliance

Motivated by this debate, we attempted to apply the AGM theory in the context of a foundational theory. Of course, there is a well-known (and easy to show) result from the literature which shows that the AGM postulates are not applicable to belief bases under the foundational model [Fuhr91], mainly because of the recovery postulate (again). However, this is true for logics that satisfy the AGM assumptions, so we chose to determine whether this fact remains true under our more liberal assumptions.

Our general approach towards this aim is the same as in the belief set case. Initially, we will reformulate the (generalized) AGM postulates, in order to capture the different semantics of the foundational case. More specifically, in the belief base context, the AGM requirement that contraction should result in a theory is dropped; furthermore, changes will be required in postulates (K–2), (K–3), as we are no longer interested in the logical closure of the KB, but on the explicit facts that appear in it only. The new, modified postulates are:

(B–1)	$K-X \subseteq L$	(*generalized base closure*)
(B–2)	$K-X \subseteq K$	(*generalized base inclusion*)
(B–3)	If $X \notin Cn(K)$, then $K-X=K$	(*generalized base vacuity*)
(B–4)	If $X \notin Cn(\emptyset)$, then $X \notin Cn(K-X)$	(*generalized base success*)
(B–5)	If $Cn(X)=Cn(Y)$, then $K-X=K-Y$	(*generalized base preservation*)
(B–6)	$K \subseteq Cn((K-X) \cup X)$	(*generalized base recovery*)

The non-generalized version of these postulates (applicable only in logics satisfying the AGM assumptions) can be found in [Fuhr91]. Notice the postulate (B−2), which captures the fact that the resulting KB should be a subset of the original KB (not of its theory).

The seemingly small differences between the "standard" and the "base" version of the postulates have severe effects on the operators considered. In fact, the base postulates are inapplicable to most logics and belief bases, due to the interaction between the base recovery and the base inclusion postulate. Here is one example:

Example 4.3. Consider that the underlying language is PL and set K={a∧b}, X={a}. The operation K−X cannot be handled in a way that satisfies the base AGM postulates. Indeed, due to (B−2), we have two options for the result (say K′=K−X): either K′=K or K′=∅. Postulate (B−4) disallows K′=K; postulate (B−6) disallows K′=∅. Thus, there can be no AGM-compliant base contraction operator that can handle this case.

The effects of this simple observation were immediate in the literature and led to the rejection of the base recovery postulate. As already explained in section 2.5, the recovery (and base recovery) postulate cannot be dropped unless replaced by other postulate(s), which would capture the intuition behind the Principle of Minimal Change. Some authors did that, by replacing the base recovery postulate with other constraints, such as filtering ([Fuhr91]). Others dropped the AGM postulates altogether, and developed a new set of postulates from scratch ([Han99b], [Han96]); this approach is reasonable, as the AGM postulates were developed with the coherence model in mind. In either case, the AGM postulates were characterized as unsuitable to handle belief base operations, even by their developers ([AGM85], [Mak87]).

Base Decomposability

To determine whether this fact remains true in logics that don't satisfy the AGM assumptions, we took a route similar to the one taken for standard AGM-compliance: we studied the logics of our generalized class and determined the conditions under which one can define a contraction operator satisfying the base AGM postulates in these logics. Not surprisingly, this study led to stronger conditions, compared to the standard case of the coherence model; furthermore, the conditions introduced for the foundational case are very similar to the ones introduced for the coherence model.

Initially, we can show that, like in the standard (coherence) case, the first five postulates can be satisfied in any logic:

Lemma 4.13. In any logic <L,Cn> there exists at least one operation that satisfies (B−1)-(B−5).

Proof

Assume the trivial function defined as:

K−X=K iff X⊄Cn(K)

K−X=∅ iff X⊆Cn(K)

It is easy to show that this function can be defined in any logic and satisfies (B−1)-(B−5). □

However, as in the standard case, the admittance of an operation that is compliant with all six postulates requires the satisfaction of certain conditions; the development of these conditions passes through the study of the effects of the base inclusion postulate in the new postulates. This postulate forces us to select a subset of the base as the result of the contraction. When contracting a belief set K in the coherence model, (K−2) limits our options

for the result to be among the subsets of Cn(K); in effect, any implication of K can be selected. When contracting a belief base K under the foundational model, (B−2) limits our options for the result drastically: only subsets of K can be selected. The key idea here is to view this additional restriction as corresponding to having a poorer underlying logic; in other words, the net effect of (B−2) is that some of the beliefs that could be selected as contraction results in the coherence case are now excluded.

In the lattice representation of a logic, we will use double circles to denote the nodes of the lattice which cannot be expressed using propositions from K alone (i.e., the nodes which are not expressible using some set in the powerset of K). These nodes cannot be taken as contraction results (in the base contraction case), even though they can be used as operands for a contraction operation (see figure 4.13).

Fuhrmann in [Fuhr91] used this observation to claim that the only bases that can satisfy the base AGM postulates are the superredundant bases, i.e., those that are closed under logical consequence (called theories in the AGM terminology [AGM85]). This claim was based on the fact that all implications of a theory (superredundant base) K are actually subsets of K; on the other hand, the powerset of a KB K does not contain all the beliefs of K's logical closure (i.e., all its implications), unless K is a theory. Thus, while the implications of a KB form a "new logic" (remember that the implications of a KB form a principal dual ideal, so they can be viewed as a lattice representing a logic), the beliefs contained in the powerset of a KB may not (unless the KB is superredundant). Worse still, even if the powerset of a KB is a "logic" in this sense, it may not satisfy the AGM assumptions for a logic, except in the special case when it is a theory (i.e., superredundant).

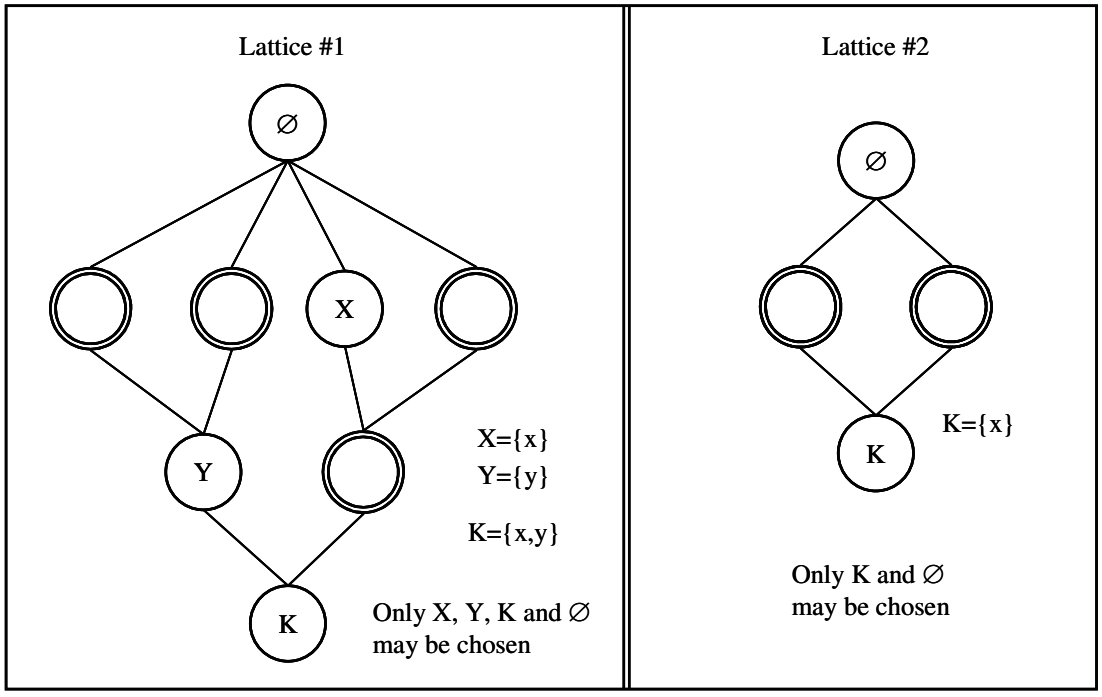

Figure 4.13. Lattices and logics (foundational case)

In a superredundant base, all nodes in the respective lattice are represented using single circles, and all can be taken as contraction results. In this special case, the compliance of an operator with the base AGM postulates coincides with ordinary AGM-compliance (see also corollary 4.15 below). But, in general, the base is not a theory, so there may be certain nodes

which are overruled, causing the two sets of postulates to differ in most cases. For example, in the first lattice of figure 4.13, only the nodes X and Y, the empty set and K itself can be taken as contraction results when K is contracted with any belief; in the second, the situation is even worse, as only K itself and the empty set can be chosen.

Given Fuhrmann's ideas, it seemed reasonable to neglect the AGM postulates when dealing with belief bases. However, there was a problem with Fuhrmann's approach: he implicitly assumed that the assumptions set by AGM were necessary for AGM-compliant operators to exist, which is not the case, as our analysis in the previous sections showed.

Despite this problem, Fuhrmann's analysis paved the way to find the conditions necessary for the existence of AGM-compliant operators for the foundational case. His syllogism can be repeated as follows: suppose a condition P that is necessary and sufficient for an AGM-compliant operator to exist in a logic (in the standard case where belief sets are considered). Assume also two sets K, X and the operation $K'=K-X$. When dealing with belief sets, we require that there exists a set K' satisfying condition P. Because of the inclusion postulate (K−2), this set can be formed using expressions from the set Cn(K), thus $K'\subseteq Cn(K)$.

In the foundational case, the syllogism is pretty much the same: we require that there exists a set K' satisfying condition P, but, the base inclusion postulate (B−2) restricts this set to be formed using elements from K only in this case (instead of Cn(K)), thus $K'\subseteq K$. Using the results of the previous sections, we can replace the condition P with the decomposability condition (definition 4.3); this gives us the desired result. To formally establish these facts we need the following definitions, which follow the general pattern used during our study of the coherence model:

Definition 4.8. An operator $-:P(L)\times P(L)\rightarrow P(L)$ is a *base-AGM-compliant contraction operator with respect to the basic base postulates for contraction* (or simply *base-AGM-compliant contraction operator*) iff it satisfies the basic base AGM postulates for contraction (B−1)-(B−6).

A logic <L,Cn> is called *base-AGM-compliant with respect to the basic postulates for contraction* (or simply *base-AGM-compliant*) iff at least one base-AGM-compliant contraction operator can be defined in this logic.

Definition 4.9. Assume any logic <L,Cn> and a set $K\subseteq L$.

- The set $K\subseteq L$ is *base decomposable* iff $X^{-}(K)\cap P(K)\neq\varnothing$ for every $X\subseteq L$
- The logic <L,Cn> is *base decomposable* iff for every K, $X\subseteq L$ $X^{-}(K)\cap P(K)\neq\varnothing$ (or equivalently iff all $K\subseteq L$ are base decomposable)

Using these definitions, the following theorem can be shown, which is analogous to theorem 4.1:

Theorem 4.4. A logic <L,Cn> is base-AGM-compliant iff it is base decomposable.

Proof

($\Rightarrow$) Suppose that the logic is base-AGM-compliant. Then there exists a base contraction function '−' that satisfies the basic base AGM postulates for contraction.

Take any K, $X\subseteq L$. If it is not the case that $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$, then either $\varnothing\in X^{-}(K)$ or $K\in X^{-}(K)$ (see the proof of lemma 4.6). But both $\varnothing$, $K\in P(K)$, so if the relation $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$ does not hold, then $X^{-}(K)\cap P(K)\neq\varnothing$.

Consider now the principal case where $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$. Set $K'=K-X$. By the base inclusion postulate we get $K'\subseteq K$, which implies that $Cn(K')\subseteq Cn(K)$ and that $K'\in P(K)$.

Suppose that $Cn(K')=Cn(K)$. Then $Cn(X)\subseteq Cn(K)=Cn(K')$, so the postulate of base success is not satisfied, a contradiction; thus, $Cn(K')\subset Cn(K)$.
By the base recovery postulate it follows that $K\subseteq Cn(K'\cup X)$, thus:
$Cn(K)\subseteq Cn(Cn(K'\cup X))=Cn(K'\cup X)\subseteq Cn(Cn(K)\cup Cn(K))=Cn(K)$, which implies that:
$Cn(K'\cup X)=Cn(K)$
We conclude that $K'\in X^{-}(K)\cap P(K)$, so $X^{-}(K)\cap P(K)\neq\emptyset$, so <L,Cn> is base decomposable.
($\Leftarrow$) Now assume that the logic is base decomposable. We define a selection function s that selects one subset of L from a collection of subsets of L. Being a selection function, it has the property that: $s(\mathcal{B})\in\mathcal{B}$ for every collection $\mathcal{B}$ of subsets of L. If $\mathcal{B}$ is empty ($\mathcal{B}=\emptyset$), then $s(\mathcal{B})$ is not defined.
We also define a function $-:P(L)\times P(L)\rightarrow P(L)$ such that:
$K-X=K$ iff $K\in X^{-}(K)$
$K-X=s(X^{-}(K)\cap P(K))$ otherwise
In short, the function '–' returns an arbitrarily selected set from $X^{-}(K)\cap P(K)$, unless K is contained in $X^{-}(K)$, in which case K is returned.
The function '–' is well-defined because $X^{-}(K)\cap P(K)\neq\emptyset$, by the hypothesis that the logic is base decomposable.
Now, we can prove that the function '–' as defined satisfies the basic base AGM postulates for contraction.
By the definition of '–', the postulate of base closure (B–1) is obviously satisfied.
Since $K-X\in X^{-}(K)\cap P(K)$, by the definition of P(K) it follows that $K-X\subseteq K$ (so (B–2) is satisfied).
If $X\not\subseteq Cn(K)$, then $X^{-}(K)=\{Y\subseteq L \mid Cn(Y)=Cn(K)\}$, thus $K\in X^{-}(K)$, so, by the definition of '–' the postulate of base vacuity (B–3) holds.
For the postulate of base success (B–4), we distinguish the different cases:

- Suppose that $Cn(\emptyset)\subset Cn(X)\subseteq Cn(K)$ and that $X\subseteq Cn(K-X)$. Since $K'\in X^{-}(K)$, we get: $Cn(K)=Cn(K'\cup X)=Cn(K')\subset Cn(K)$, a contradiction.
- In all other cases we get that $K\in X^{-}(K)$; thus: $K'=K$. Suppose, for the sake of contradiction, that $X\not\subseteq Cn(\emptyset)$ and $X\subseteq Cn(K')$. Then $X\subseteq Cn(K)$, thus $Cn(\emptyset)\subset Cn(X)\subseteq Cn(K)$, a contradiction by the hypothesis regarding X and K.

Thus, the postulate of base success (B–4) holds.
For the postulate of base preservation (B–5), take two $X, Y\subseteq L$ such that $Cn(X)=Cn(Y)$. Then, by lemma 4.2, it follows that $X^{-}(K)=Y^{-}(K)$, so $K-X=K-Y$.
For the postulate of base recovery (B–6), we will again distinguish the different cases:

- If $Cn(\emptyset)\subset Cn(X)\subseteq Cn(K)$, then, since $K'\in X^{-}(K)$, we get, by the definition of $X^{-}(K)$, that: $K\subseteq Cn(K)=Cn(K'\cup X)$ by the definition of $X^{-}(K)$.
- In any other case: $Cn(K'\cup X)=Cn(K\cup X)\supseteq K$, using similar arguments.

So (B–6) holds.
We conclude that '–' is a base-AGM-compliant contraction operator, so <L,Cn> is base-AGM-compliant. □

We can also get the following corollary:
Corollary 4.12. A logic <L,Cn> is base-AGM-compliant iff for all K, $X\subseteq L$ with $Cn(\emptyset)\subset Cn(X)\subset Cn(K)$ there exists a $K'\subseteq L$ such that $Cn(K'\cup X)=Cn(K)$, $Cn(K')\subset Cn(K)$ and $K'\subset K$. Equivalently, the logic <L,Cn> is base-AGM-compliant iff for all K, $X\subseteq L$ with $Cn(\emptyset)\subset Cn(X)\subset Cn(K)$ it holds that $X^{-}(K)\cap P(K)\neq\emptyset$.

Notice that the only difference between decomposability and base decomposability has to do with the range of the selection of the set K′: in decomposability, K′ must be selected from the set of complement beliefs of the contracted expression X with respect to K ($X^{-}(K)$); in base decomposability, K′ must *additionally* be a subset of K ($K' \subseteq K$). The fact that $Cn(K') \subset Cn(K)$, by the definition of $X^{-}(K)$, allows us to reduce the latter condition to the one appearing in corollary 4.12 ($K' \subset K$). In other words, a set K is base decomposable if each of its proper implications has a "complement" relative to K that can be expressed using propositions in K only (and not Cn(K) as was the case with simple decomposability). In the first lattice of figure 4.13, K is base decomposable (and decomposable); on the other hand, in the second lattice, K is not base decomposable, even though it is decomposable. This observation indicates that base decomposability is a stronger condition than decomposability. This is not surprising, as (B−2) is stronger than (K−2):

Corollary 4.13. Consider a logic <L,Cn>. The following hold:

- If the logic <L,Cn> is base decomposable, then it is decomposable.
- If the logic <L,Cn> is base-AGM-compliant, then it is AGM-compliant.
- If a set $K \subseteq L$ is base decomposable, then it is decomposable.

Base Cuts

The close connection between decomposability and base decomposability could imply a similar connection between cuts and a similar structure for bases, the *base cuts*. Indeed, such a connection exists, as shown in the following definition and theorem:

Definition 4.10. Consider a logic <L,Cn>, a belief $K \subseteq L$ and a set of beliefs $\wp \subseteq P(L)$ such that:

- For all $Y \in \wp$, $Cn(Y) \subset Cn(K)$
- For all $Z \subseteq L$ such that $Cn(Z) \subset Cn(K)$ and $Z \subset K$, there is a $Y \in \wp$ such that $Cn(Y) \subseteq Cn(Z)$ or $Cn(Z) \subseteq Cn(Y)$

Then $\wp$ is called a *base cut* of A.

Base cuts divide only the subsets of K instead of the subsets of Cn(K). Thus, a cut is a base cut, but not vice-versa. The figure below shows two different base cuts:

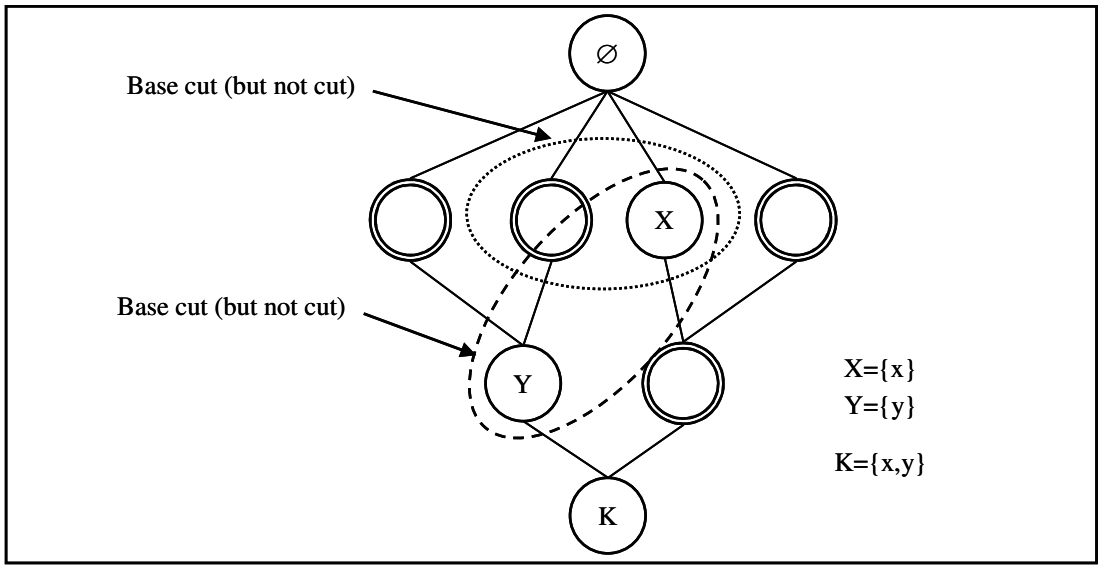

Figure 4.14. Examples of base cuts

The counterpart of theorem 4.2 for belief base operations is:

Theorem 4.5. Consider a logic <L,Cn> and a set K⊆L. Then K is base decomposable iff for all base cuts $\wp$ of K it is the case that $Cn(\cap_{Y\in\wp}Cn(Y))=Cn(\varnothing)$.

Proof

(⇒) Suppose that K is base decomposable.

If $Cn(K)=Cn(\varnothing)$ then there is only one base cut of K (namely $\wp=\varnothing$), so the theorem holds trivially.

Suppose that $Cn(K)\neq Cn(\varnothing)$ and that for some base cut $\wp$ of K it holds that $Cn(\cap_{Y\in\wp}Cn(Y))=X\neq Cn(\varnothing)$. Obviously it holds that $X=Cn(X)\subseteq Cn(Y)$ for all $Y\in\wp$ thus $Cn(X)\subset Cn(K)$.

We will show that $X^{-}(K)\cap P(K)=\varnothing$.

Take any $Z\in X^{-}(K)\cap P(K)$. Then $Z\subseteq K$.

If Z=K then Cn(Z)=Cn(K) thus $Z\notin X^{-}(K)$, a contradiction; thus, $Z\subset K$.

Since $Z\in X^{-}(K)$, it holds that $Cn(Z)\subset Cn(K)$. By the definition of the base cut, there exists a $Y\in\wp$ such that $Cn(Y)\subseteq Cn(Z)$ or $Cn(Z)\subseteq Cn(Y)$.

If $Cn(Y)\subseteq Cn(Z)$ then:

$Cn(X)\subseteq Cn(Y)\subseteq Cn(Z)$, so: $Cn(X\cup Z)=Cn(Z)\subset Cn(K) \Rightarrow Z\notin X^{-}(K)$, a contradiction.

If $Cn(Z)\subseteq Cn(Y)$ then:

$Cn(Z)\subseteq Cn(Y)$ and $Cn(X)\subseteq Cn(Y)$, so: $Cn(X\cup Z)\subseteq Cn(Cn(Y)\cup Cn(Y))=Cn(Y)\subset Cn(K) \Rightarrow$

$\Rightarrow Z\notin X^{-}(K)$, a contradiction.

We conclude that $X^{-}(K)\cap P(K)=\varnothing$, so K is not base decomposable, a contradiction.

Thus, for any cut $\wp$: $Cn(\cap_{Y\in\wp}Cn(Y))=Cn(\varnothing)$.

(⇐) Now suppose that K is not base decomposable. Then by lemma 4.6, there exists a set $X\subseteq L$ such that $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$ and $X^{-}(K)\cap P(K)=\varnothing$. We will construct a base cut $\wp$ such that $Cn(\cap_{Y\in\wp}Cn(Y))\neq Cn(\varnothing)$.

Take any $Z\subseteq L$ such that $Z\in P(K)$ and $Cn(Z)\subset Cn(K)$. There is at least one such set, for example $Z=\varnothing$.

Then: $Cn(Z\cup X)\subseteq Cn(Cn(K)\cup Cn(K))=Cn(K)$.

If $Cn(Z\cup X)=Cn(K)$ then $Z\in X^{-}(K)\cap P(K)$, a contradiction because $X^{-}(K)\cap P(K)=\varnothing$.

So $Cn(Z\cup X)\subset Cn(K)$ for all $Z\in P(K)$ with $Cn(Z)\subset Cn(K)$.

Take the family of beliefs: $\wp=\{Z\cup X \mid \text{for all } Z\subseteq L \text{ such that } Cn(Z)\subset Cn(K) \text{ and } Z\subset K\}$.

For all $Y\in\wp$: $Cn(Y)\subset Cn(K)$ by the previous result.

Furthermore, for any $Z'\subseteq L$ such that $Z'\subset K$ and $Cn(Z')\subset Cn(K)$ it holds that $Cn(Z')\subseteq Cn(Z'\cup X)$ and $Z'\cup X\in\wp$.

Thus, $\wp$ is a base cut. Furthermore, for all $Y\in\wp$: $Cn(Y)\supseteq Cn(X)$, therefore:

$Cn(\cap_{Y\in\wp}Cn(Y))\supseteq Cn(X)\supset Cn(\varnothing)$.

So, we have constructed a base cut with the desired property and the proof is complete. □

The following corollary is immediate:

Corollary 4.14. A logic <L,Cn> is base-AGM-compliant iff for all K⊆L and all base cuts $\wp$ of K it is the case that $Cn(\cap_{Y\in\wp}Cn(Y))=Cn(\varnothing)$.

At this point it would be anticipated (and desirable) that the analogous of the notion of max-cuts for the foundational case could be defined (i.e., the notion of a base-max-cut). Unfortunately, the different nature of the foundational case makes the counterpart of a max-cut problematic in this context; for this reason, base-max-cuts are not studied in this thesis.

Discussion on Base-AGM-Compliance

Our analysis opens up a different viewpoint on the connection between the foundational model and the AGM theory. Up to now, it was considered that the AGM postulates were incompatible with belief base operations because there are no such operators that satisfy them. Our analysis implies that the AGM postulates are incompatible with respect to belief base operations when applied to non-base-AGM-compliant logics (such as the logics considered in the AGM framework); this is similar to the coherence case, where the AGM postulates are inapplicable to non-AGM-compliant logics.

As a first example, we will show that there are logics, not in the class AGM considered, that are base-AGM-compliant:

Example 4.4. Take the logic of example 4.2 (see also figure 4.12 in page 106):

$L=\{x_1, x_2, \ldots, x_n\}$

$Cn(\varnothing)=\varnothing$

$Cn(\{x_i\})=\{x_i\}$, for all i=1,2,…,n

$Cn(X)=L$ for all X with more than one element ($|X|\geq 2$)

This logic was shown to not satisfy the AGM assumptions and to be AGM-compliant (see example 4.2). To show that the logic is base-AGM-compliant as well, take any K, $X\subseteq L$ such that $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$.

By the definition of the logic, the latter condition implies that $X=\{x\}$ for some $x\in L$, $|K|\geq 2$ and $Cn(K)=L$.

Since $|K|\geq 2$, there are two y, $z\in K$ such that $y\neq z$; if $y=x$ then it holds that $\{z\}\in X^{-}(K)\cap P(K)$; in any other case, $\{y\}\in X^{-}(K)\cap P(K)$. These facts follow easily from the logic's definition.

In any case, $X^{-}(K)\cap P(K)\neq\varnothing$, so the logic is base-AGM-compliant. Like in example 4.2, the result can be generalized to the case where L is infinite (countable or uncountable).

Example 4.4 implies that there are several (finite and infinite) logics that are base-AGM-compliant. In a base-AGM-compliant logic all sets, even non-superredundant, are base decomposable by definition. So, it should not be assumed beforehand that an AGM-compliant operator for non-superredundant belief bases is impossible.

Another connection between decomposability, base decomposability and superredundant KBs is given in the following simple lemma:

Lemma 4.14. Consider a logic <L,Cn> and any decomposable set $K\subseteq L$. If $K=Cn(K)$, then K is base decomposable.

Proof

Take any $X\subseteq L$. Then $X^{-}(K)\neq\varnothing$. For any $Y\in X^{-}(K)$, $Y\subseteq Cn(Y)\subset Cn(K)=K$, so $Y\in P(K)$.

In conclusion, $X^{-}(K)\cap P(K)=X^{-}(K)\neq\varnothing$. □

Lemma 4.14 implies that all theories of an AGM-compliant logic are base decomposable; this is actually the claim made by Fuhrmann in [Fuhr91], because logics satisfying the AGM assumptions are AGM-compliant (corollary 4.9). The following corollary is based on lemma 4.14 and provides an alternative method to check base decomposability, applicable in certain special cases:

Corollary 4.15. Consider an AGM-compliant logic <L,Cn>. If $K=Cn(K)$ for all $X\subseteq L$, then <L,Cn> is base-AGM-compliant.

The main problem with base-AGM-compliance is that most interesting logics are actually non-base-AGM-compliant. As the following proposition shows, base decomposability is a

very strong property satisfied only by a few uninteresting logics (which are not part of the AGM framework):

Proposition 4.9. Consider a logic <L,Cn>. If there is a proposition $x \in L$ and a set $Y \subseteq L$ such that $Cn(\varnothing) \subset Cn(Y) \subset Cn(\{x\})$, then this logic is not base-AGM-compliant.

Proof

Set $K=\{x\}$. Then $P(K)=\{\varnothing,K\} \Rightarrow Y^{-}(K) \cap P(K) \subseteq \{\varnothing,K\}$.

By the definition of $Y^{-}(K)$, $K \notin Y^{-}(K)$; on the other hand, $Cn(\varnothing \cup Y)=Cn(Y) \subset Cn(K)$, so $\varnothing \notin Y^{-}(K)$. Thus, $Y^{-}(K) \cap P(K)=\varnothing$, so the logic is not base-AGM-compliant. □

This result is very strong: every logic in which there is a proposition implying any other non-tautological proposition is not base-AGM-compliant. The contraction example $\{a \wedge b\}-\{b\}$ (example 4.3) presented earlier in this section was an application of this general pattern. This immediately disallows all logics of the AGM framework, as well as most logics useful for knowledge representation (such as DLs, as will be shown later):

Corollary 4.16. All logics satisfying the AGM assumptions are not base-AGM-compliant.

On the other hand, the fact that a given logic is not base decomposable does not rule out the possibility that this logic contains base decomposable beliefs, which may or may not be theories (superredundant). The following example provides evidence for this fact:

Example 4.5. Consider PL limited to only two atoms a, b, as well as the set $K=\{a \vee b, a \vee \neg b\}$. Obviously, K is not a theory (e.g., $a \in Cn(K) \backslash K$). The only $X \subseteq L$ for which $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$ holds are the sets $\{a \vee b\}$ and $\{a \vee \neg b\}$ (see also figure 4.5 in page 95). We can easily show that both these sets have a complement relative to K which is a subset of K (namely $\{a \vee \neg b\}$ and $\{a \vee b\}$ respectively); thus, K is base decomposable.

4.9. Related Results and Discussion

Roots

To shed more light on the intuition behind AGM-compliance (and base AGM-compliance), we will study a relevant concept, the concept of *roots* of a logic. Roots represent the weakest possible interesting beliefs that a logic can express. These beliefs are very important for AGM-compliance.

Take any AGM-compliant logic <L,Cn> and any set $X \subseteq L$. As mentioned before, decomposability implies that, if there is a set X_1 such that $Cn(\varnothing) \subset Cn(X_1) \subset Cn(X)$ then there is also another set X_2 such that $Cn(X_2) \subset Cn(X)$ and $Cn(X_1 \cup X_2)=Cn(X)$. In a sense, X is "decomposed" in two "smaller" sets X_1, X_2. Subsequently, both X_1 and X_2 can be "decomposed" in X_{11}, X_{12}, X_{21}, X_{22} and so on in a recursive manner. This procedure may continue indefinitely or it may ultimately stop at a set (belief) which has no further implications (except $Cn(\varnothing)$ and itself, of course). For finite logics, all such decompositions will ultimately reach some point where no further decomposition is possible, because finiteness guarantees that we will eventually run out of sets; for infinite logics, some of the decompositions may stop and some may continue indefinitely. In either case, sets which cannot be further decomposed play a major role in our theory and are called *roots* of the logic (see also figure 4.15):

Definition 4.11. Consider a logic <L,Cn>. A set $X \subseteq L$ is called a *root* of the logic iff $Cn(X) \neq Cn(\varnothing)$ and there is no $Y \subseteq L$ such that $Cn(\varnothing) \subset Cn(Y) \subset Cn(X)$.

Notice that the definition of a root is independent of the AGM-compliance of the logic; furthermore, not all logics contain roots. When they exist though, roots are both decomposable and base decomposable:

Lemma 4.15. Any root in any logic is base decomposable (thus decomposable).

Proof

Obvious by the definition of roots. □

Furthermore, roots can always be expressed using some singular set:

Lemma 4.16. Consider a logic <L,Cn>. For any root $X \subseteq L$ there is some $x \in L$ such that $X \cong \{x\}$.

Proof

There is at least one $x \in X$ such that $Cn(\{x\}) \neq Cn(\varnothing)$; in a different case $Cn(X)=Cn(\varnothing)$ so X would not be a root.

For this particular x, it obviously holds that $Cn(\{x\}) \subseteq Cn(X)$.

If $Cn(\{x\}) \subset Cn(X)$, then X would not be a root, thus $Cn(\{x\})=Cn(X)$ and the proof is complete. □

The following notion is closely related to roots; it is also interesting for our study of AGM-compliance (see also figure 4.15):

Definition 4.12. A logic <L,Cn> *contains a fully independent root set* iff there exists a family of beliefs $\wp=\{Y_i \subseteq L, i \in I\}$ such that:

- For all $X \subseteq L$ there exists a set of indices $I(X) \subseteq I$ such that $Cn(\cup_{i \in I(X)} Y_i)=Cn(X)$
- For all $j \in I$, $Y_j \notin Cn(\cup_{i \in I \setminus \{j\}} Y_i)$

The family $\wp=\{Y_i \mid i \in I\}$ is called a *fully independent root set* of <L,Cn>.

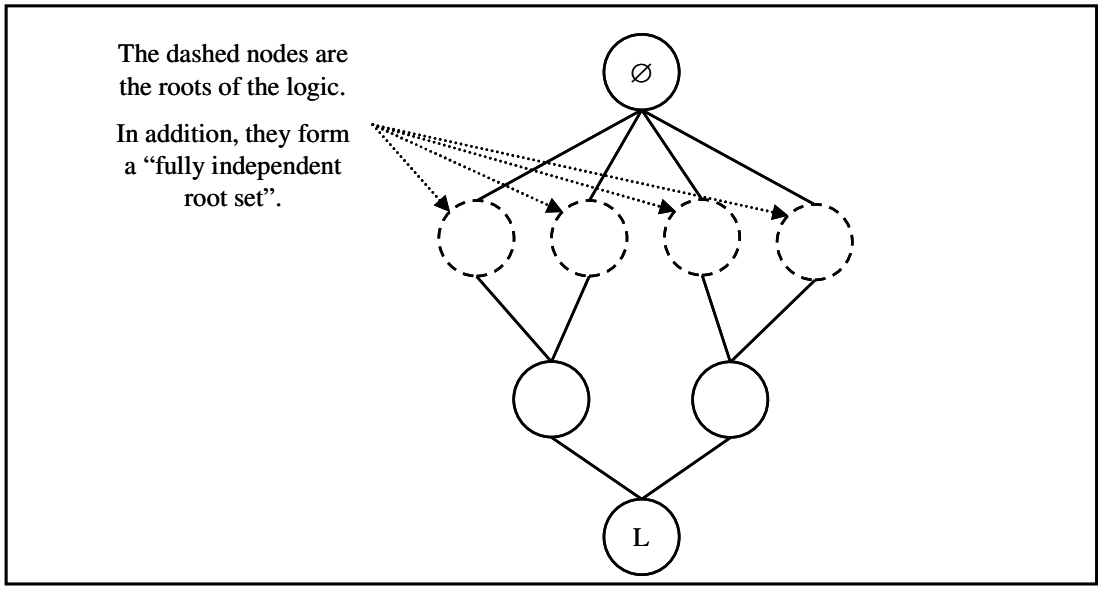

Figure 4.15. Roots and fully independent root sets

The following lemma shows that any belief in logics which contain a fully independent root set is completely characterized by the members of the fully independent root set that it implies:

Lemma 4.17. Consider a logic <L,Cn> that contains a fully independent root set $\wp=\{Y_i \mid i \in I\}$. For any $X \subseteq L$ there exists a unique set $I(X) \subseteq I$ such that $Cn(\cup_{i \in I(X)} Y_i)=Cn(X)$. Furthermore, $I(X)=\{i \in I \mid X \vdash Y_i, Y_i \in \wp\}$.

Proof
The existence of the set I(X) follows from the definition.
For uniqueness, suppose that there exists a second set $I(X)' \subseteq I$ such that:
$Cn(X)=Cn(\cup_{i\in I(X)}Y_i)=Cn(\cup_{i\in I(X)'}Y_i)$.
Take any $j\in I(X)\backslash I(X)'$. Then:
$Y_j \subseteq Cn(X)=Cn(\cup_{i\in I(X)'}Y_i)\subseteq Cn(\cup_{i\in I\backslash\{j\}}Y_i)$, a contradiction by the definition.
Thus $I(X)\backslash I(X)'=\varnothing$.
Similarly, we can show that $I(X)'\backslash I(X)=\varnothing$, which implies that $I(X)=I(X)'$.
For the second part of the lemma, take any $j\in I(X)$.
Then $Cn(X)=Cn(\cup_{i\in I(X)}Y_i)\supseteq Cn(Y_j) \Rightarrow X\vdash Y_j$.
Thus, $I(X)\subseteq\{i\in I \mid X\vdash Y_i, Y_i\in\wp\}$.
Now take any $j\in\{i\in I \mid X\vdash Y_i, Y_i\in\wp\}$. Suppose that $j\notin I(X)$. Then:
$Y_j \subseteq Cn(X)=Cn(\cup_{i\in I(X)}Y_i)\subseteq Cn(\cup_{i\in I\backslash\{j\}}Y_i)$, a contradiction by the definition. So $j\in I(X)$, thus $\{i\in I \mid X\vdash Y_i, Y_i\in\wp\}\subseteq I(X)$. We conclude that $I(X)=\{i\in I \mid X\vdash Y_i, Y_i\in\wp\}$. □

The importance of a fully independent root set in a logic that contains such a set, is emphasized by the fact that even the inference relation (equivalently the consequence operator) can be characterized using this set:
Lemma 4.18. Consider a logic <L,Cn> that contains a fully independent root set $\wp=\{Y_i \mid i\in I\}$ and two sets $X, Y\subseteq L$. Then $Y\vdash X$ iff $I(X)\subseteq I(Y)$.
Proof
($\Rightarrow$) Take any $Y_i\in\wp$ such that $X\vdash Y_i$.
Since $Y\vdash X$ we get that $Y\vdash Y_i$.
By lemma 4.14: $I(X)=\{i\in I \mid X\vdash Y_i, Y_i\in\wp\}$.
Combining these facts: $I(X)\subseteq I(Y)$.
($\Leftarrow$) If $I(X)\subseteq I(Y)$ then $Cn(X)=Cn(\cup_{i\in I(X)}Cn(Y_i))\subseteq Cn(\cup_{i\in I(Y)}Cn(Y_i))=Cn(Y)$, so $Y\vdash X$. □

The following lemma uncovers some more properties of a logic that contains a fully independent root set:
Lemma 4.19. Consider a logic <L,Cn> that contains a fully independent root set $\wp=\{Y_i \mid i\in I\}$. Then the following hold:

- For all $j\in I$, $I(Y_j)=\{j\}$
- For all $j, k\in I$, $j\neq k$, $Cn(Y_j)\cap Cn(Y_k)=Cn(\varnothing)$
- If $Cn(X)\subset Cn(Y_j)$ for some $j\in I$, then $Cn(X)=Cn(\varnothing)$
- For all $i\in I$, there exists a $x\in L$ such that $Cn(\{x\})=Cn(Y_i)$

Proof
The first result is obvious by lemma 4.18.
For the second, set $Z=Cn(Y_j)\cap Cn(Y_k)$.
Then there is a set $I(Z)\subseteq I$ such that $Cn(Z)=Cn(\cup_{i\in I(Z)}Y_i)=Cn(Y_j)\cap Cn(Y_k)$.
But $Cn(Z)\subseteq Cn(Y_j) \Rightarrow I(Z)\subseteq I(Y_j)=\{j\}$ and $Cn(Z)\subseteq Cn(Y_k) \Rightarrow I(Z)\subseteq I(Y_k)=\{k\}$.
So, $I(Z)\subseteq\{j\}\cap\{k\}=\varnothing$, thus $Z=Cn(Z)=Cn(\varnothing)$.
For the third result, suppose that $Cn(X)\subset Cn(Y_j)$ for some $j\in I$.
Then, by lemma 4.18, $I(X)\subset I(Y_j)=\{j\}$, thus $I(X)=\varnothing \Rightarrow Cn(X)=Cn(\varnothing)$.
For the fourth result, take any $i\in I$ and set $Y=Cn(Y_i)$.
If $Y=\varnothing$ then $Y_i=\varnothing\subseteq Cn(\cup_{j\in I\backslash\{i\}}Y_j)$, a contradiction, even if $I=\varnothing$.
So $Y\neq\varnothing$.
If for all $x\in Y$ it holds that $Cn(\{x\})=Cn(\varnothing)$ then $Cn(Y)=Cn(\varnothing)$, thus:

$Y_i \subseteq Cn(Y_i)=Cn(\varnothing)\subseteq Cn(\cup_{j\in I\setminus\{i\}} Y_j)$, a contradiction.
So there is at least one $x\in Y$ such that $Cn(\{x\})\supset Cn(\varnothing)$.
If, for this $x\in Y$, it holds that $Cn(\{x\})\subset Cn(Y)$ then $Cn(\varnothing)\subset Cn(\{x\})\subset Cn(Y)$, a contradiction by the third result of this lemma.
Thus $Cn(\{x\})=Cn(Y)$. □

As already mentioned, there is a close connection between roots and fully independent root sets. The following lemma shows this connection; a fully independent root set (if it exists) is comprised by roots only; moreover, all roots have an equivalent belief in the fully independent root set:

Lemma 4.20. Consider a logic <L,Cn> that contains a fully independent root set $\wp=\{Y_i \mid i\in I\}$ and the family of sets $\wp_0=\{X\subseteq L \mid X \text{ is a root of the logic}\}$. Then the following hold:

- $\wp\subseteq\wp_0$
- For any $X\in\wp_0$ there exists a $Y_j\in\wp$ such that $Cn(X)=Cn(Y_j)$

Proof

The third result of lemma 4.19 implies that $\wp\subseteq\wp_0$.
For the second bullet, take any $X\in\wp_0$.
There is a set $I(X)\subseteq I$ such that $Cn(\cup_{i\in I(X)} Y_i)=Cn(X)$.
$I(X)\neq\varnothing$ because $Cn(X)\neq Cn(\varnothing)$.
Take any $j, k\in I(X)$ and suppose that $j\neq k$. Then:
$I(Y_j)=\{j\}\subset\{j,k\}\subseteq I(X)$, so $\varnothing\subset I(Y_j)\subset I(X)$, thus $Cn(\varnothing)\subset Cn(Y_j)\subset Cn(X)$, a contradiction because X is a root ($X\in\wp_0$).
So $j=k$, thus $I(X)=\{j\}$ and $Cn(X)=Cn(Y_j)$. □

These results imply the following interesting lemma:

Lemma 4.21. Consider a logic <L,Cn> that contains a fully independent root set $\wp=\{Y_i \mid i\in I\}$. Then, there is a fully independent root set $\wp'=\{Y_i' \mid i\in I'\}$, such that all Y_i' are singular sets (i.e., of the form $Y_i'=\{y_i\}$).

Proof

By the fourth bullet of lemma 4.19, for any $Y_i\in\wp$ there is some $y_i\in L$ such that $Y_i\cong\{y_i\}$. Set $\wp'=\{\{y_i\} \mid i\in I\}$. Since for all $i\in I$ it holds that $\{y_i\}\cong Y_i$ it is easy to show that $\wp'$ is a fully independent root set too. □

Our results show that the roots of a logic are indeed the weakest pieces of non-trivial (i.e., non-tautological) information that the logic can express (the most vague information). Some logics are completely characterized by their roots, in the sense that every belief of the logic is uniquely characterized by the roots it implies; this way, the whole structure of the logic can be determined by its roots. Such logics contain a fully independent root set. In a logic that contains a fully independent root set, any belief can be broken down in its roots (in the sense that it is equivalent to the union of the roots it implies) and no root is implied by any combination of the other roots. All roots, as well as the members of a fully independent root set, can be expressed using single elements from the logic, i.e., using singular sets.

Logics that have the structure implied by the existence of a fully independent root set possess interesting properties, such as AGM-compliance:

Proposition 4.10. Consider a logic <L,Cn> that contains a fully independent root set $\wp=\{Y_i \mid i\in I\}$. Then <L,Cn> is AGM-compliant.

Proof

Take any K, X⊆L such that Cn(∅)⊂Cn(X)⊂Cn(K). By lemma 4.18, we conclude that ∅⊂I(X)⊂I(K). Set I(Z)=I(K)\I(X) and Z=Cn($\cup_{i\in I(Z)}Y_i$).
Then: I(Z)⊂I(K) ⇒ Cn(Z)⊂Cn(K).
Moreover: Cn(X∪Z)=Cn(Cn(X)∪Cn(Z))=Cn(Cn($\cup_{i\in I(X)}Y_i$)∪Cn($\cup_{i\in I(Z)}Y_i$))=
=Cn(($\cup_{i\in I(X)}Y_i$)∪($\cup_{i\in I(Z)}Y_i$))=Cn($\cup_{i\in I(K)}Y_i$)=Cn(K).
So Z∈X⁻(K) ⇒ X⁻(K)≠∅, so the logic is AGM-compliant. □

The proof of this proposition also implies a scheme for defining one possible AGM-compliant contraction operator in such a logic: to define the operation K−X, take the roots that are implied by K, but not implied by X, and calculate their union. The result is a member of the set of complement beliefs of X with respect to K (X⁻(K)).

Notice that all the other methods that we have for showing AGM-compliance (decomposability, cuts and max-cuts) require the determination of the decomposability for all sets that are definable in the logic (i.e., for all K⊆L). With roots, we don't need to check every such set; if a fully independent root set exists, then all sets in the logic are decomposable and the logic is AGM-compliant. However, this characterization is not complete, i.e., not all AGM-compliant logics contain a fully independent root set.

The structure forced by a fully independent root set in the lattice representing a logic can sometimes only appear "locally", but not "globally". In such cases, there is no fully independent root set for the logic (lattice), but a similar structure appears for a certain principal dual ideal of the lattice (logic) formed by the consequences of a given set K⊆L. This situation appears when a set K is the only set of its equivalence class, i.e., there is no set K′≠K that is equivalent to K. Initially, let us show that such sets (and all their subsets) are decomposable:

Lemma 4.22. Consider a logic <L,Cn> and a belief K⊆L such that: Cn(K)=Cn(K′) iff K=K′. Then, any set K_0⊆K is decomposable.

Proof

By the hypothesis, by setting K′=Cn(K), we get that K=Cn(K).
Now take any K_0⊆K and a X⊆L such that Cn(∅)⊂Cn(X)⊂Cn(K_0). Set Y=K_0\X.
Obviously Cn(Y∪X)=Cn((K_0\X)∪X)=Cn(K_0).
Moreover Cn(Y)⊆Cn(K_0). Suppose that Cn(Y)=Cn(K_0).
Then: Cn(K)=Cn((K\$K_0$)∪$K_0$)=Cn((K\$K_0$)∪Y) ⇒ K=K\$K_0$∪Y.
The latter relation implies that Y⊇K_0.
Combining this fact with the definition of Y we get that X=∅, so Cn(X)=Cn(∅), a contradiction.
We conclude that Cn(Y)⊂Cn(K_0), which means that Y∈X⁻(K_0).
This implies that K_0 is decomposable, and the proof is complete. □

As already mentioned, sets with this property force a structure similar to the one forced by a fully independent root set. This is shown by the fact that, in the special case where K=L in lemma 4.22, we get that the logic contains a fully independent root set:

Proposition 4.11. Consider a logic <L,Cn> such that Cn(X)=L iff X=L. Then, the logic contains a fully independent root set.

Proof

Suppose that L={x_i | i∈I} for some set I.
Set ℘={{x_i} | i∈I}, i.e., the set of all the singular sets which are subsets of L.
Then for all X⊆L there is a I(X)={j | x_j∈X}⊆I such that Cn($\cup_{i\in I(X)}${x_i})=Cn(X).

Moreover, if for some $j \in I$ it holds that $\{x_j\} \subseteq Cn(\cup_{i \in I \setminus \{j\}} \{x_i\})$ then:
$Cn(\cup_{i \in I \setminus \{j\}} \{x_i\}) = Cn(\cup_{i \in I \setminus \{j\}} \{x_i\} \cup \{x_j\}) = Cn(\cup_{i \in I} \{x_i\}) = L$ which implies that:
$\cup_{i \in I \setminus \{j\}} \{x_i\} = L$, an absurdity.
Thus $\wp$ is a fully independent root set. □

Using either lemma 4.22 or proposition 4.11 we can show the following interesting corollary:

Corollary 4.17. Consider a logic <L,Cn> such that $Cn(X)=L$ iff $X=L$. Then, <L,Cn> is AGM-compliant.

Notice that corollary 4.17 gives another (non-complete) way to show that a logic is AGM-compliant without having to determine the decomposability of all beliefs definable in the logic.

The above results show the connection of roots and fully independent root sets with AGM-compliance. Similar connections can be shown for the case of base-AGM-compliance. The following proposition shows that, if a logic contains a fully independent root set, then any set that contains its roots is base decomposable:

Proposition 4.12. Consider a logic <L,Cn> that contains a fully independent root set $\wp = \{Y_i \mid i \in I\}$ and any $K \subseteq L$. If for any $Y_i \in \wp$ such that $K \vdash Y_i$ there is some root Y_i' such that $Cn(Y_i) = Cn(Y_i')$ and $Y_i' \subseteq K$, then K is base decomposable.

Proof
Since <L,Cn> contains a fully independent root set, <L,Cn> is AGM-compliant by proposition 4.10, so K is decomposable.
Take any $X \subseteq L$ such that $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$. Then $X^{-}(K) \neq \varnothing$.
Consider some $Z \in X^{-}(K)$. Then $Cn(Z) \subset Cn(K)$, so $I(Z) \subset I(K)$.
For all $i \in I(Z) \subset I(K)$, $K \vdash Y_i$ so there is some root Y_i' such that $Cn(Y_i) = Cn(Y_i')$ and $Y_i' \subseteq K$.
Set $Z_0 = \cup_{i \in I(Z)} Y_i'$.
Then $Z_0 \subseteq K$, by the hypothesis.
Moreover $Cn(Z_0) = Cn(\cup_{i \in I(Z)} Y_i') = Cn(\cup_{i \in I(Z)} Y_i) = Cn(Z)$.
Thus:
$Cn(Z_0) = Cn(Z) \subset Cn(K)$
$Cn(Z_0 \cup X) = Cn(Z \cup X) = Cn(K)$.
Therefore $Z_0 \in X^{-}(K) \cap P(K)$, so K is base decomposable. □

But that is not all; here is an interesting consequence of proposition 4.12:

Proposition 4.13. Consider a logic <L,Cn> that contains a fully independent root set $\wp = \{Y_i \mid i \in I\}$. If for all $x \in L$ it holds that $Cn(\{x\}) = Cn(\varnothing)$ or $\{x\}$ is a root, then <L,Cn> is base-AGM-compliant.

Proof
Take any $K \subseteq L$ and any $x \in K$.
If $\{x\}$ is a root, then by lemma 4.20, there is some $Y_i \in \wp$ such that $Y_i \cong \{x\}$.
Set $I' = \{i \in I \mid$ there is some $x \in K$ such that $Y_i \cong \{x\}\}$.
Set $K' = \{x \in K \mid Cn(\{x\}) \neq Cn(\varnothing)\}$.
Obviously $K' = \{x \in K \mid \{x\}$ is a root$\}$.
By the hypothesis, it follows that $Cn(K) = Cn(K') = Cn(\cup_{j \in I'} Cn(Y_i))$
Now take any $Y_i \in \wp$ such that $K \vdash Y_i$.
Suppose that there is no $x \in K$ such that $\{x\} \cong Y_i$.

Then $i \notin I'$ and:
$Cn(Y_i) \subseteq Cn(K) = Cn(\cup_{j \in I'} Cn(Y_i)) \subseteq Cn(\cup_{j \in I \setminus \{i\}} Y_i)$.
This is a contradiction by the definition of a fully independent root set.
Thus, for all $Y_i \in \wp$ such that $K \vdash Y_i$ there is a $x \in K$ such that $\{x\} \cong Y_i$ and $\{x\}$ is a root.
Since $\{x\} \subseteq K$, proposition 4.12 implies that <L,Cn> is base-AGM-compliant. □

Combining these results, the following proposition can be shown:

Proposition 4.14. Consider a logic <L,Cn> that contains a fully independent root set $\wp = \{Y_i \mid i \in I\}$. Then, <L,Cn> is base-AGM-compliant iff for all $x \in L$ it holds that $Cn(\{x\}) = Cn(\varnothing)$ or $\{x\}$ is a root.

Proof

($\Rightarrow$) Suppose that <L,Cn> is base-AGM-compliant and take any $x \in L$ such that $Cn(\{x\}) \neq Cn(\varnothing)$. If $\{x\}$ is not a root then there is some $Y \subseteq L$ such that $Cn(\varnothing) \subset Cn(Y) \subset Cn(X)$, so by proposition 4.9, <L,Cn> is not base-AGM-compliant, a contradiction.
($\Leftarrow$) The opposite is immediate by proposition 4.13. □

The latter result provides an alternative equivalent method to check for base-AGM-compliance, applicable only in logics which contain a fully independent root set. Furthermore, the combination of proposition 4.11 with proposition 4.13, gives the following corollary, which is a stronger version of corollary 4.17:

Corollary 4.18. Consider a logic <L,Cn> such that $Cn(X) = L$ iff $X = L$. Then the logic is base-AGM-compliant.

Proof

By proposition 4.11, the logic contains a fully independent root set. If there are two propositions $x, y \in L$ such that $\{x\} \vdash \{y\}$, then $Cn(L \setminus \{y\}) = Cn(L) = L$, a contradiction. So every belief of the form $\{x\}$ is a root. Thus, by proposition 4.13, we get that <L,Cn> is base-AGM-compliant. □

These results show the numerous connections between roots, singular sets, the concept of a fully independent root set, AGM-compliance and base-AGM-compliance. The concept of roots is quite instructive on the intuition behind decomposition (and base decomposition). It uncovers the recursive character of decomposition by providing a base for the recursion. Notice that a base for this recursion does not always exist, even in decomposable logics, because, in infinite logics, such recursion may continue indefinitely.

We believe that the structure of roots determines, to some extent, the properties of a logic with respect to AGM-compliance. In this respect, fully independent root sets may not be the only interesting topology of roots in a logic; a further study on roots may uncover other root topologies related to AGM-compliance. Therefore, future research on roots (and fully independent root sets) may provide alternative characterizations of decomposable and base decomposable logics, more sophisticated than the ones appearing in this subsection.

Supplementary Postulates

As already mentioned in chapter 2, AGM introduced, in addition to the six basic postulates, two supplementary ones which were not considered in our initial study of AGM-compliance [AGM85]. In this subsection, we will study these postulates and show that, in many cases, the form of AGM-compliance introduced in definition 4.1 is strong enough to include the supplementary postulates as well.

As usual, our first step is to provide an adequate reformulation of the postulates (K−7) and (K−8). This can be easily made as follows:

(K−7) $(K-X)\cap(K-Y)\subseteq K-(X\cup Y)$

(K−8) If $X\notin K-(X\cup Y)$ then $K-(X\cup Y)\subseteq K-X$

As with the basic postulates, it is trivial to show that, in the presence of the AGM assumptions each of the original (K−7), (K−8) (see section 2.5) is equivalent to its generalized version. Furthermore, it can be shown that the trivial operator used in the proof of lemma 4.1 satisfies (K−7), (K−8) as well:

Lemma 4.23. In any logic <L,Cn> there exists at least one operation that satisfies (K−1)-(K−5), (K−7), (K−8).

Proof

Consider the following function:

$K-X=Cn(K)$ iff $X\notin Cn(K)$

$K-X=Cn(\varnothing)$ iff $X\subseteq Cn(K)$

It is easy to show that this function can be defined in any logic and satisfies (K−1)-(K−5) (see also lemma 4.1).

For (K−7), (K−8), the result can be shown by distinguishing between the different cases:

If $X\subseteq Cn(K)$ and $Y\subseteq Cn(K)$ then $X\cup Y\subseteq Cn(K)$ so:

$K-X=Cn(\varnothing)$, $K-Y=Cn(\varnothing)$ and:

$K-(X\cup Y)=Cn(\varnothing)$, so both (K−7) and (K−8) hold.

If $X\subseteq Cn(K)$ and $Y\notin Cn(K)$ then $X\cup Y\notin Cn(K)$, so:

$K-X=Cn(\varnothing)$, $K-Y=Cn(K)$ and:

$K-(X\cup Y)=Cn(K)$.

But then $X\subseteq K-(X\cup Y)=Cn(K)$, so both (K−7) and (K−8) hold.

If $X\notin Cn(K)$ and $Y\subseteq Cn(K)$ then $X\cup Y\notin Cn(K)$, so:

$K-X=Cn(K)$, $K-Y=Cn(\varnothing)$ and:

$K-(X\cup Y)=Cn(K)$, so both (K−7) and (K−8) hold.

If $X\notin Cn(K)$ and $Y\notin Cn(K)$ then $X\cup Y\notin Cn(K)$ so:

$K-X=Cn(K)$, $K-Y=Cn(K)$ and:

$K-(X\cup Y)=Cn(K)$, so both (K−7) and (K−8) hold. □

This lemma is a generalization of lemma 4.1. Of course, once the recovery postulate is added to our list of desired postulates, not all logics admit an operator satisfying all eight postulates (not even the six basic ones, as we showed in example 4.1). The similarities with the standard case do not stop here; it can be shown that under certain assumptions, decomposability is a necessary and sufficient condition for the existence of a function that satisfies all eight postulates (not just the six basic ones). Before showing this, we will define the notion of full AGM-compliance:

Definition 4.13. An operator $-:P(L)\times P(L)\rightarrow P(L)$ is an *AGM-compliant contraction operator with respect to the basic and supplementary postulates for contraction* (or simply *fully AGM-compliant contraction operator*) iff it satisfies the basic and the supplementary AGM postulates for contraction (K−1)-(K−8) .

A logic <L,Cn> is called *AGM-compliant with respect to the basic and supplementary postulates for contraction* (or simply *fully AGM-compliant*) iff at least one fully AGM-compliant contraction operator can be defined in this logic.

Our main result will require one lemma:

Lemma 4.24. Consider a logic <L,Cn> and three sets K, X, Y⊆L with the following properties:

- K has a max-cut (say $\wp$)
- Cn(∅)⊂Cn(X)⊆Cn(K)
- Cn(∅)⊂Cn(Y)⊆Cn(K)

Then, the following hold:

- $X^-(K)\cap\wp\neq\emptyset$ iff $X^-(K)\neq\emptyset$
- $(X\cup Y)^-(K)\cap\wp = (X^-(K)\cap\wp) \cup (Y^-(K)\cap\wp)$

Proof

If $X^-(K)\cap\wp\neq\emptyset$, then, obviously, $X^-(K)\neq\emptyset$.

For the opposite, if $X^-(K)\neq\emptyset$, then, by the definition of $X^-(K)$ and the definition of a max-cut, for any $Z\in X^-(K)$ there is some $Z'\in\wp$ such that $Cn(Z)\subseteq Cn(Z')\subset Cn(K)$.

By lemma 4.5, this implies that $Z'\in X^-(K)$, so $X^-(K)\cap\wp\neq\emptyset$.

For the second result, take any $Z\in(X\cup Y)^-(K)\cap\wp$.

Then:

$Z\in\wp$

$Cn(Z)\subset Cn(K)$

Since $Cn(Z)\subset Cn(K)$ and $Cn(X)\subseteq Cn(K)$, we get: $Cn(Z\cup X)\subseteq Cn(K)$

Similarly: $Cn(Z\cup Y)\subseteq Cn(K)$

Assume that $Z\notin X^-(K)\cup Y^-(K)$

Then $Z\notin X^-(K)$ and $Z\notin Y^-(K)$, so $Cn(Z\cup X)\subset Cn(K)$ and $Cn(Z\cup Y)\subset Cn(K)$.

By the definition of a max-cut, there is some $Z'\in\wp$ such that $Cn(Z')\supseteq Cn(Z\cup X)\supseteq Cn(Z)$ and $Z\in\wp$. Thus, $Z'=Z$, so $Cn(Z\cup X)=Cn(Z)$.

Similarly, we can show that $Cn(Z\cup Y)=Cn(Z)$.

Thus:

$Cn(K)=Cn(Z\cup X\cup Y)=Cn((Z\cup X)\cup(Z\cup Y))=Cn(Cn(Z\cup X)\cup Cn(Z\cup Y))=$

$=Cn(Cn(Z)\cup Cn(Z))=Cn(Z)$, a contradiction.

Thus $Z\in X^-(K)\cup Y^-(K)$.

If $Z\in X^-(K)$, then, since $Z\in\wp$, we conclude that $Z\in X^-(K)\cap\wp$.

If $Z\in Y^-(K)$, then, since $Z\in\wp$, we conclude that $Z\in Y^-(K)\cap\wp$.

Thus:

$(X\cup Y)^-(K)\cap\wp \subseteq (X^-(K)\cap\wp) \cup (Y^-(K)\cap\wp)$.

Now consider a $Z\in(X^-(K)\cap\wp) \cup (Y^-(K)\cap\wp)$.

Obviously: $Cn(\emptyset)\subset Cn(X)\subseteq Cn(X\cup Y)\subseteq Cn(K)$, so, by lemma 4.3:

$X^-(K)\subseteq(X\cup Y)^-(K) \Rightarrow X^-(K)\cap\wp\subseteq(X\cup Y)^-(K)\cap\wp$.

Similarly: $Y^-(K)\subseteq(X\cup Y)^-(K) \Rightarrow Y^-(K)\cap\wp\subseteq(X\cup Y)^-(K)\cap\wp$.

Thus:

$(X^-(K)\cap\wp) \cup (Y^-(K)\cap\wp) \subseteq (X\cup Y)^-(K)\cap\wp$ and the proof is complete. □

Using this lemma, the following proposition can be shown:

Proposition 4.15. Consider a logic <L,Cn> with the property that all beliefs K⊆L have a max-cut. Then <L,Cn> is fully AGM-compliant iff it is AGM-compliant.

Proof

The (⇒) direction is obvious.

For the opposite consider any two sets K, X⊆L, and set $\wp$ the max-cut of K.

By the Axiom of Choice (see [End77], [RR85]) we can define a well-order in $\wp$, i.e., a full ordering $\leq$ of $\wp$ with a minimal element for all $\wp' \subseteq \wp$.
We will use the notation $\min(\wp')$ to denote the minimal element of $\wp'$ under the above ordering.
For the same reason, we can define a similar well-order in $X^-(K)$ and denote with $\min(\mathcal{E})$ the minimal element of a set $\mathcal{E} \subseteq X^-(K)$.
With these tools we can define the function $-:P(L)\times P(L)\rightarrow P(L)$ such that:
$K-X=Cn(\min(X^-(K)\cap\wp))$ iff $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$
$K-X=Cn(\min(X^-(K)))$ in a different case
Since <L,Cn> is AGM-compliant, $X^-(K)\neq\varnothing$, and, by lemma 4.24, $X^-(K)\cap\wp\neq\varnothing$, when $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$.
Thus, the function '–' is well defined in any case.
We will show that it is also fully AGM-compliant.
By corollary 4.1, the function '–' is AGM-compliant; it remains to show that it satisfies (K–7) and (K–8).
To do that, consider another $Y\subseteq L$. Set:
$Z=X\cup Y$
$X'=K-X$
$Y'=K-Y$
$Z'=K-Z$
We must distinguish several different cases for the ordering between X, Y, Z and K:
Suppose that: $Cn(X)=Cn(\varnothing)$. Then, by the definition of $X^-(K)$ and '–': $X'=Cn(K)$.
Furthermore, $Cn(Z)=Cn(Y)$, so $Z^-(K)=Y^-(K)$, so $Z'=Y'$.
Using these identities, we can easily verify that (K–7), (K–8) hold.
Similarly, if $Cn(Y)=Cn(\varnothing)$, then $Y'=Cn(K)$ and $X'=Z'$; using these identities we can easily verify that (K–7), (K–8) hold.
Suppose that $Cn(X)\neq Cn(\varnothing)$ and $Cn(X)\not\subseteq Cn(K)$. Then $X'=Cn(K)$ again.
Also, $Cn(Z)\not\subseteq Cn(K)$, so $Z'=Cn(K)$.
Using these identities, we can easily verify that (K–7), (K–8) hold.
Suppose that $Cn(\varnothing)\subset Cn(X)\subseteq Cn(K)$ and $Cn(Y)\neq Cn(\varnothing)$ and $Cn(Y)\not\subseteq Cn(K)$.
Then $Y'=Cn(K)$ and $Cn(Z)\not\subseteq Cn(K)$, so $Z'=Cn(K)$.
Using these identities, we can easily verify that (K–7), (K–8) hold.
The only case left to consider is when $Cn(\varnothing)\subset Cn(X)\subseteq Cn(K)$ and $Cn(\varnothing)\subset Cn(Y)\subseteq Cn(K)$.
In this case, it also holds that $Cn(\varnothing)\subset Cn(Z)\subseteq Cn(K)$.
There is some Z_0 such that $Cn(Z_0)=Z'$ and $Z_0=\min(Z^-(K)\cap\wp)$.
By lemma 4.24, it holds that: $Z^-(K)\cap\wp = (X^-(K)\cap\wp) \cup (Y^-(K)\cap\wp)$.
Since $Z_0\in Z^-(K)\cap\wp$, we conclude that $Z_0\in X^-(K)\cap\wp$ or $Z_0\in Y^-(K)\cap\wp$.
If $Z_0\in X^-(K)\cap\wp$, then, given that $X^-(K)\cap\wp\subseteq(X\cup Y)^-(K)\cap\wp$ and Z_0 is the minimal element of $Z^-(K)$, we can conclude that Z_0 is also the minimal element of $X^-(K)\cap\wp$.
Thus: $X'=Cn(Z_0)=Z'$
Thus: $(K-X)\cap(K-Y)=X'\cap Y'\subseteq X'=Z'=K-(X\cup Y)$, so (K–7) holds.
Also $K-(X\cup Y)=Z'=X'=K-X$ so (K–8) holds as well.
Now suppose that $Z_0\in Y^-(K)\cap\wp$.
If we suppose, additionally, that $Z_0\in X^-(K)\cap\wp$ as well, then by the previous case, we are done. So we are forced to assume that $Z_0\notin X^-(K)\cap\wp$.
Using similar syllogisms as in the previous case we conclude that: $Y'=Z'$.

Therefore:
$(K-X)\cap(K-Y)=X'\cap Y'\subseteq Y'=Z'=K-(X\cup Y)$, so (K−7) holds.
Suppose that $X\not\subseteq Z'$.
Then $Cn(X)\not\subseteq Z'=Cn(Z')$, thus $Cn(X\cup Z')\supset Cn(Z')$.
But $Cn(X)\subseteq Cn(K)$, $Cn(Z')\subseteq Cn(K)$, so $Cn(X\cup Z')\subseteq Cn(K)$
If $Cn(X\cup Z')\subset Cn(K)$ then, since $\wp$ is a max-cut, there is some $Z''\in\wp$ such that: $Cn(K)\supset Cn(Z'')\supseteq Cn(X\cup Z')\supset Cn(Z')$, and, since $Z'\in\wp$, we conclude that $Z'=Z''$. Thus: $Cn(X\cup Z')=Cn(Z')$, a contradiction.
Thus: $Cn(X\cup Z')=Cn(K)$; but then, given that $Cn(Z')\subset Cn(K)$, we conclude that $Z'\in X^{-}(K)$; since $Cn(Z_0)=Z'$, we get that $Z_0\in X^{-}(K)$ and $Z_0\in\wp$, so $Z_0\in X^{-}(K)\cap\wp$, a contradiction by our initial hypothesis.
In conclusion: $X\subseteq Z'=K-(X\cup Y)$, so (K−8) holds in a trivial way.
We have shown that, in all cases, the function '−' satisfies (K−7), (K−8), so '−' is fully AGM-compliant, and the proof is complete. □

The latter result shows that, in many cases, AGM-compliance and full AGM-compliance coincide. Combined with corollary 4.6, this fact implies that the addition of supplementary postulates to our list of desirable postulates does not rule out AGM-compliant logics with a finite number of equivalence classes:
Corollary 4.19. Consider any logic <L,Cn> with the property that $P(L)/\cong$ is finite. Then, this logic is AGM-compliant iff it is fully AGM-compliant.

Unfortunately, we have been unable to extend the result of proposition 4.15 to be applicable in the infinite case (i.e., when a max-cut cannot be defined). The main difficulty with the supplementary postulates is that they have a more "global" character than the basic ones. One can define K−X independently for each K, X if he is interested in satisfying the six basic postulates only; but when the supplementary postulates are taken into account, the result of an operation K−X affects (directly or indirectly) the results of the operations K−Y for all Y that are implied by K, so one unifying scheme should be defined in order to determine the result of K−X for each K, X. The determination of the exact relation between AGM-compliance and full AGM-compliance in the general case is part of our future work.

Generalized AGM Postulates for Revision

Apart from contraction, AGM studied another important operation, revision [AGM85]. As already mentioned in section 2.5, eight postulates (six basic and two supplementary) were defined for revision as well. To provide a generalized theory of revision, we will perform the same basic steps as in the case of contraction. As we will see however, when it comes to the AGM theory of revision, important complications arise from the very start.

First, let us review the postulates. The first five basic postulates can be easily reformulated as follows:

(K+1) $K+X=Cn(K+X)$
(K+2) $X\subseteq K+X$
(K+3) If $Cn(K\cup X)\neq L$ then $K+X=Cn(K\cup X)$
(K+4) If $Cn(X)\neq L$, then $Cn(K+X)\neq L$
(K+5) If $Cn(X)=Cn(Y)$, then $K+X=K+Y$

This reformulation is not as trivial as it was in the case of contraction. The term "consistent" has been interpreted as "not implying the whole L", so the consequences of a

consistent set X are not the whole language L ($Cn(X)\neq L$). The clause "$\neg x\notin Cn(\varnothing)$" has been replaced by $Cn(X)\neq L$; this is the same intuitively, because if the negation of a proposition (in the classical case) is a tautology, then the proposition itself is a contradiction, so its consequences are the whole language (as above). The clause "$\neg x\notin Cn(K)$" denotes that the negation of x is not an implication of K; in the classical setting, this is intuitively the same as saying that K does not contradict x, i.e., that $K\cup\{x\}$ is not contradictory; therefore, the clause "$\neg x\notin Cn(K)$" has been replaced by the clause $Cn(K\cup X)\neq L$.

The aforementioned tricks allow us to generalize the postulates without using the notion of negation, which is generally missing in logics under Tarski's framework. As in the case of contraction, it is easy to show that each of the generalized (K+1)-(K+5) is equivalent to its original counterpart (K+1)-(K+5) under the AGM assumptions. Similarly, each of the original supplementary postulates (K+7), (K+8) can be reformulated using the same tricks; like before, each of these postulates is equivalent, in the presence of the AGM assumptions, to its generalized counterpart that appears below:

(K+7) $K+(X\cup Y)\subseteq Cn((K+X)\cup Y)$

(K+8) If $Cn((K+X)\cup Y)\neq L$, then $Cn((K+X)\cup Y)\subseteq K+(X\cup Y)$

We left the sixth postulate (K+6) for last, because its generalization poses serious problems. (K+6) is a special postulate, in the sense that it involves both operations (contraction and revision). Here is the original version of the postulate:

(K+6) $(K+x)\cap K = K-(\neg x)$

A straightforward generalization of (K+6), using the concept of the negation of a belief, should be something like:

(K+6a) $(K+X)\cap K = K-(\neg X)$

This postulate however is ill-formed; as already mentioned in section 4.7, the negation of a belief X cannot be defined in all logics of Tarski's framework and, even when definable, it is not always unique. Unfortunately, in the case of (K+6), we cannot get rid of the negation using any simple trick like the ones employed before; instead, we must dig deep into the intuitions that led to the introduction of this postulate.

Recall that (K+6) is an alternative formulation of the Harper identity [Gar92a]:

$$K-x=(K+\neg x)\cap K$$

The intuition behind this identity has been discussed in section 2.5. In short, (K+6), or, equivalently, the Harper identity, says that the part of the original KB (K) that remains in the revised KB (K+x) should be the same as the result of the contraction of K with the negation of the revising expression ($K-(\neg x)$).

According to [Mak87], (K+6) is not a "postulate" in the classical sense; in the same paper, three different readings of the postulate are provided. Under the first reading, (K+6) provides a method to define a contraction operation from a given revision operation (this is the viewpoint of the Harper identity). Under the second reading, (K+6) is an existential postulate: the revision operator should be such that there is some contraction operator (satisfying the contraction postulates) for which the relation $(K+x)\cap K=K-(\neg x)$ holds. The third reading is, in a sense, the opposite of the first: the revision operator is understood as being defined from a given contraction operator; under this reading, the revision operator is an operator that satisfies (K+6) only, while (K+1)-(K+5) are understood as derived facts for this operator.

Here, we propose a fourth reading of the postulate, related to the Principle of Minimal Change. In certain cases, some of the knowledge of the original KB should be dropped in order to accommodate the new knowledge (revision) consistently. However, the only revision postulate that constrains the "loss of information" is (K+3), which deals with the limiting case when the new knowledge is consistent with the old one; in this case, no loss of information is necessary, so (K+3) constrains the result to be equal to $Cn(K \cup X)$, in effect constraining the loss of information to be minimal (i.e., zero in this case).

But what happens when K and X are contradictory? In that case, at least *some* information from the original KB should be dropped and this information loss should be minimal (see the "expected" result in each of the cases of figure 4.16). In this sense, (K+6) is an attempt to "restrict" the information loss indirectly, through the operation of contraction in which the Principle of Minimal Change has been directly encoded in the postulates (K−3) and (K−6). More specifically, (K+6) constrains the information loss to be exactly the same as the one that is retracted when we contract $\neg X$ from the KB. The latter "information loss" is well-encoded through the contraction postulates, so why devise a new postulate to encode this principle in revision and not use the already performed work on contraction instead?

Unfortunately, none of these understandings of (K+6) leads to any straightforward formulation of the postulate to apply in Tarski's model. All four of them use the concept of "negation" in the standard, classical sense; yet, in our context, we cannot generally express the negation of a belief X. For logics and beliefs X where there is no negation of X, there is no belief, or combination of beliefs, that captures exactly the notion of the "complement of X" (i.e., negation of X). For logics and beliefs X where there are several negations, none of those beliefs captures exactly the notion of a complement either; each of these "negations" is a partial formulation of X's complement. Unfortunately, no combination of these negations would result in the "negation of X" in all cases, because there is simply no such thing in certain logics under Tarski's model. But even when the negation is unique, the structure of the logic could be so exotic that a "negation" does not have the standard intuitive meaning (Y is a "negation" of X in the second lattice of figure 4.11, page 103, but it does not capture the meaning intended here). Defining a postulate that would work only in the limited case when negation is unique and has the standard intuitive meaning is unacceptable, because such an option would undermine one of our main objectives, namely the unconstrained generalization of the AGM theory in all logics under the framework of Tarski.

These considerations imply that we should devise some notion that would lead to a set (belief) to replace $\neg X$ in the formulation of (K+6a). This notion should be definable in all logics under Tarski's model. Unfortunately, our research has not yet led to the definition of a notion that captures this intuition in an adequate manner. In fact, we are not sure whether such a notion actually exists. A final answer on this issue is one of the objectives of our future work. In the rest of this subsection, we will present some preliminary, mostly informal, ideas towards a proper definition of this notion, as well as some thoughts towards an alternative formulation of (K+6a) that would avoid the definition of a negation altogether. This discussion will allow us to develop certain intuitions that could ultimately lead to the final answer.

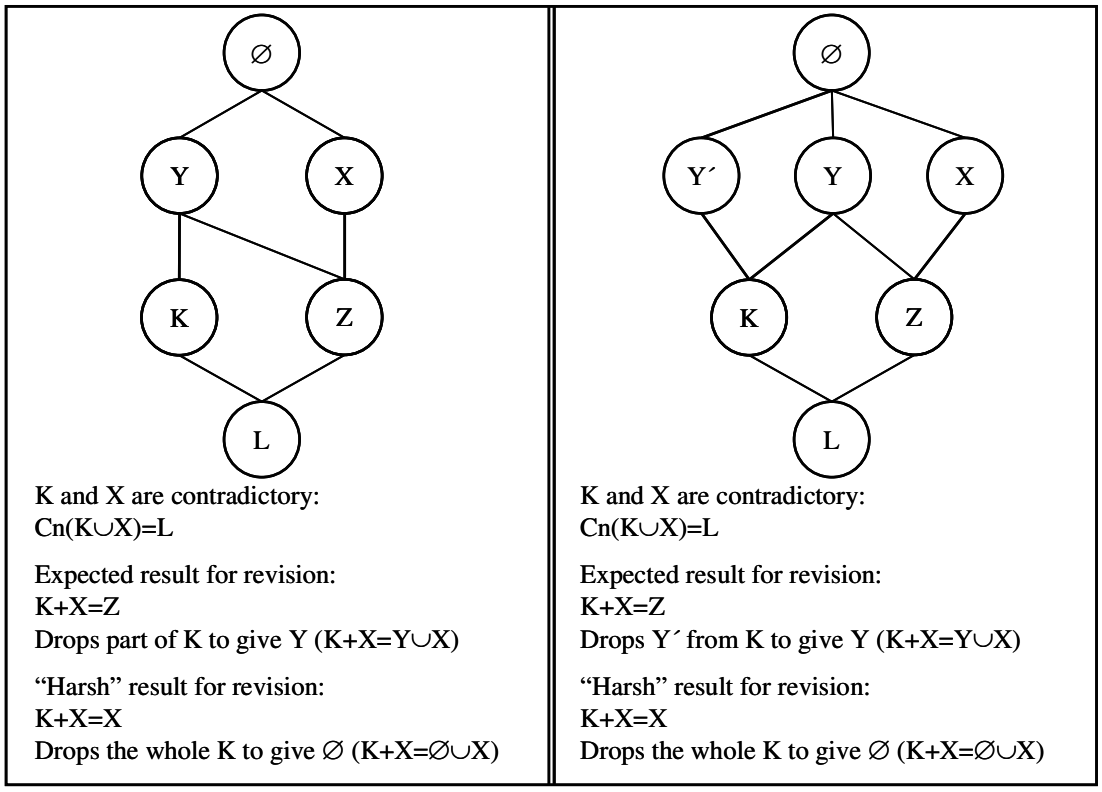

Figure 4.16. Revision example

The simplest way to deal with this problem is to remove this postulate altogether from our list of postulates. This option is favored by the analysis performed in [Mak87], in which the only understanding under which (K+6) becomes a postulate in the "real" sense is the second one, under which it postulates the existence of a certain contraction function satisfying it. However, under this understanding, (K+6) becomes a redundant postulate (in the classical setting), being derivable from (K+1)-(K+5); all we have to do is define '−' using the Harper identity and verify (using (K+1)-(K+5)) that '−' so defined satisfies (K−1)-(K−6), so it is an AGM-compliant contraction operator [Mak87]. All these hold in the AGM setting, of course; the problem is that we cannot verify these facts in the general setting without providing a generalized version of (K+6) first.

On the other hand, there are important theoretical reasons against this simple option. The most important one is the following lemma:

Lemma 4.25. Consider a logic <L,Cn>. There is some operator '+' satisfying (K+1)-(K+5) and (K+7), (K+8).

Proof

Take two beliefs K, X⊆L and set:

K+X=Cn(K∪X) iff Cn(K∪X)≠L

K+X=Cn(X) iff Cn(K∪X)=L

We will show that '+' satisfies the postulates (K+1)-(K+5) and (K+7), (K+8).

(K+1) is obviously satisfied.

For (K+2), X⊆Cn(X)⊆Cn(K∪X), so (K+2) is satisfied.

(K+3) is obvious by the definition of '+'.

Suppose that Cn(X)≠L and Cn(K+X)=L.

If Cn(K∪X)≠L then K+X=Cn(K∪X)≠L, a contradiction.

If $Cn(K \cup X)=L$ then $K+X=Cn(X) \neq L$ by our assumption, a contradiction again.
So (K+4) holds.
If $Cn(X)=Cn(Y)$ then $Cn(K \cup X)=Cn(K \cup Y)$ so (K+5) is also satisfied.
For (K+7):
If $Cn(K \cup X \cup Y) \neq L$ then $Cn(K \cup X) \neq L$ so:
$K+(X \cup Y)=Cn(K \cup X \cup Y)$ and:
$Cn((K+X) \cup Y)=Cn(Cn(K \cup X) \cup Y)=Cn(K \cup X \cup Y)=K+(X \cup Y)$
If $Cn(K \cup X \cup Y)=L$ then
$Cn((K+X) \cup Y) \supseteq Cn(Cn(X) \cup Y)=Cn(X \cup Y)=K+(X \cup Y)$
Thus, (K+7) holds.
For (K+8):
If $Cn(K \cup X \cup Y) \neq L$ then $Cn(K \cup X) \neq L$ so:
$K+(X \cup Y)=Cn(K \cup X \cup Y)$ and:
$Cn((K+X) \cup Y)=Cn(Cn(K \cup X) \cup Y)=Cn(K \cup X \cup Y)=K+(X \cup Y)$
If $Cn(K \cup X \cup Y)=L$ and $Cn(K \cup X)=L$ then:
$K+(X \cup Y)=Cn(X \cup Y)$
$Cn((K+X) \cup Y)=Cn(Cn(X) \cup Y)=Cn(X \cup Y)=K+(X \cup Y)$
If $Cn(K \cup X \cup Y)=L$ and $Cn(K \cup X) \neq L$ then:
$Cn((K+X) \cup Y)=Cn(K \cup X \cup Y)=L$, so the premise $Cn((K+X) \cup Y) \neq L$ is not true.
Thus, in any case, (K+8) holds, and the proof is complete. □

The importance of this lemma is two-fold. First, it shows that, just like in the case of contraction, there is an operator that satisfies all basic and supplementary postulates, except the problematic one (K+6), in all logics of our framework. This means that if we finally choose to drop (K+6), our study of AGM-compliance with respect to the revision postulates is already over, as all logics admit such an operator.

The second important point implied by the lemma (in fact, by the proof of the lemma) is that there are rather trivial operators which satisfy the remaining seven postulates. The operator that was used in the proof of the lemma is very simple; it is also rather "harsh" on the KB in the sense that, whenever K and X contradict each other, we drop K altogether and keep X. This is not what we normally do: we don't drop all of our beliefs whenever a new information contradicts the old; we just drop those that are required to avoid introducing a contradiction (see figure 4.16). The dropped beliefs should be as few as possible, according to the Principle of Minimal Change. This shows that we cannot generally ignore (K+6) in a theory of rational revisions, unless we replace it with some other postulate.

An alternative to (K+6) would be to assume that there is some subset Y of Cn(K), which does not contradict X and is "maximal" in some sense. Said Y would replace the clause "$K-(\neg X)$" in (K+6a). Allowing Y to be any subset of Cn(K) would lead to the same problems as in lemma 4.25, so Y should maximize some quantity. This brings us to the main problem with this alternative, i.e., how to define "maximality" and what quantity to maximize. Set-theoretic maximization would not work, because we cannot always find a maximal set that is consistent with X (for the same reasons that there is not always a max-cut of a set, see figure 4.10 and figure 4.17).

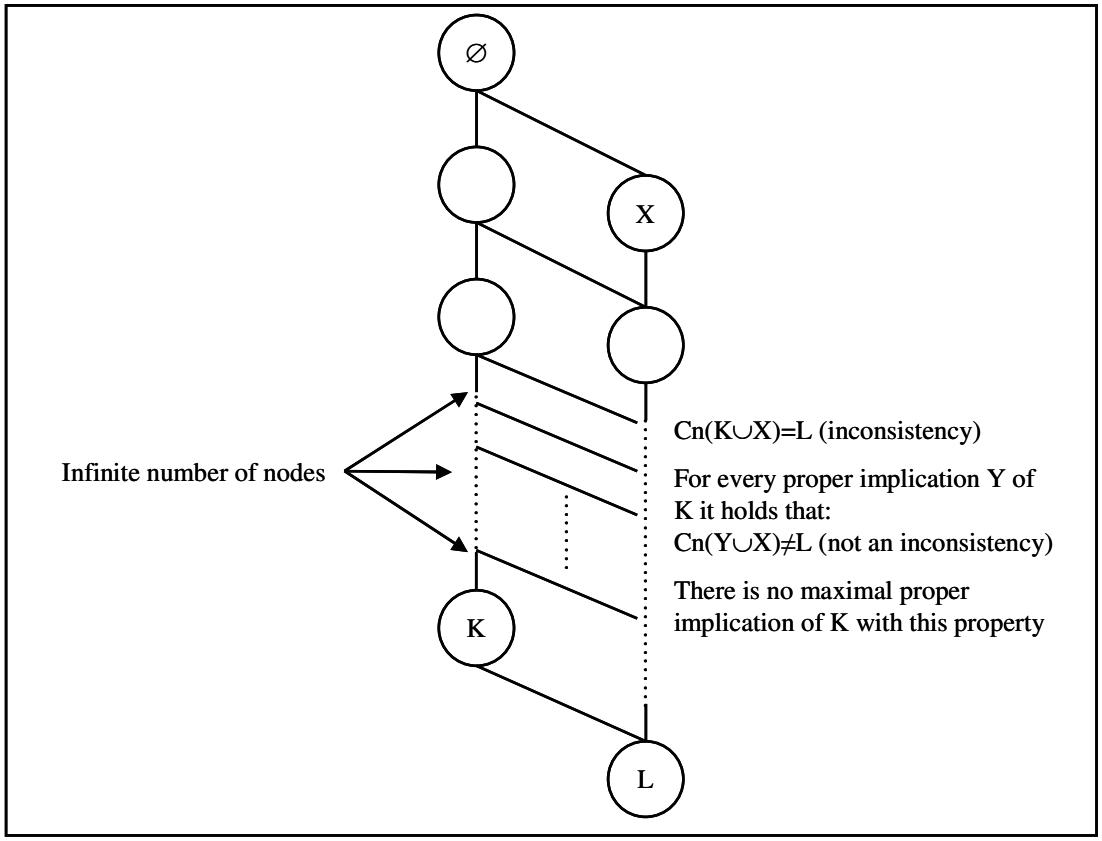

Figure 4.17. Maximality in revision

A further alternative for (K+6a) would be to slightly reformulate the postulate to take into account the fact that $\neg X$ is not always unique. There are two approaches here. According to the first, we require the existence of *some* negation of X which satisfies the postulate; this way the postulate becomes:

(K+6b) There is some negation of X (say Y) for which: $(K+X)\cap K = K-Y$

The problem with the existential approach is that whenever no negation exists, no revision operator that satisfies (K+6b) can be defined.

The second approach requires that the postulate is satisfied *for all* negations of X. In this case the postulate becomes:

(K+6c) For all negations Y of X it holds that: $(K+X)\cap K = K-Y$

The problem with the universal approach is that whenever more than one negation exists, there is no guarantee that the contraction of K by each of these negations will result to the same KB. If there are two negations of X (say Y_1, Y_2) for which $K-Y_1 \neq K-Y_2$ holds, then we are doomed: no revision operator satisfying (K+6c) can be defined.

The considerations presented provide certain propositions for generalizing the thorny (K+6) postulate. Unfortunately, no satisfactory answer to this issue has been provided; the focus of this thesis lies mainly on the contraction operator which is the most important operator for theoretical purposes [AGM85]. Nevertheless, the revision operator is an important operator, so the study of the generalization of the AGM theory of revision is an important part of our future work.

Topology of AGM-Compliance

This discussion completes the picture regarding the relation of the class of logics satisfying the AGM assumptions, the AGM-compliant and base-AGM-compliant logics and the logics under Tarski's model. This relation is shown in the following figure:

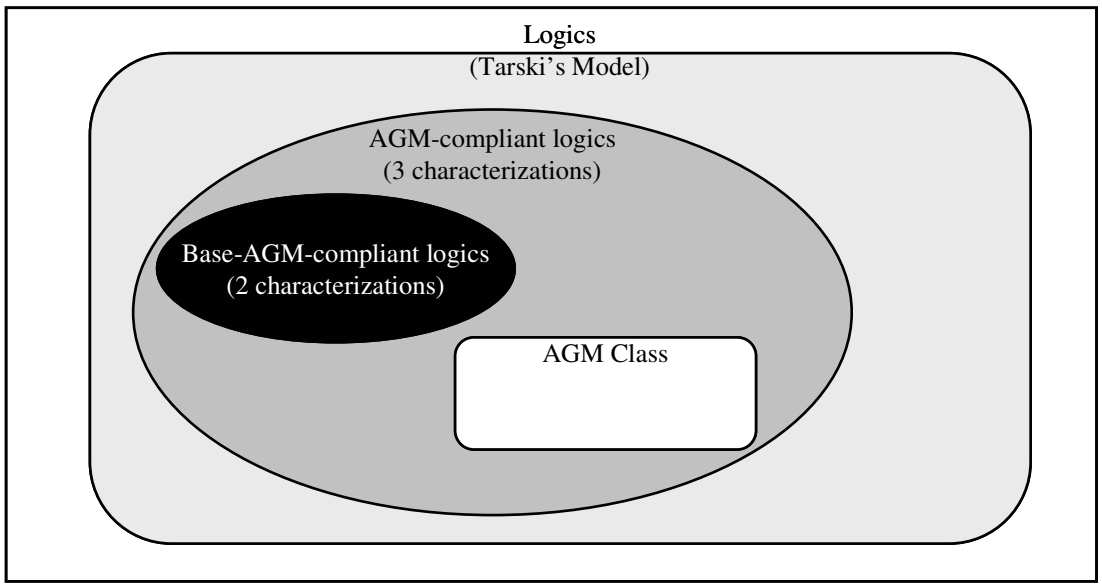

Figure 4.18. Topology of AGM-compliance

As we can see in this figure, the class of AGM-compliant logics lies somewhere between the class AGM considered and the class describable by an <L,Cn> pair, under Tarski's model (see example 4.1 and example 4.2); the class of base-AGM-compliant logics is a proper subclass of AGM-compliant logics (corollary 4.13) and has an empty intersection with the AGM class (corollary 4.16). Three equivalent characterizations of AGM-compliance have been provided (theorem 4.1, theorem 4.2, theorem 4.3) and two equivalent characterizations for base-AGM-compliance (theorem 4.4 and theorem 4.5). Fully AGM-compliant logics form a subclass of AGM-compliant logics (obviously), but we are not yet certain whether it is a proper subclass or not, except in a special case (proposition 4.15); for this reason, this class is not shown in the figure.

Chapter 5: DLs, OWL and the AGM Theory

If you're not failing every now and again, it's a sign you're not doing anything very innovative.

Woody Allen

5.1. Introduction

In the previous chapter, we set the foundations upon which the study of the applicability of the AGM theory in various logics can be based. This study was placed in a very general setting, being applicable to several logics, including, but not limited to, logics commonly used in the Semantic Web. In this chapter, we apply these results to determine the AGM-compliance of two families of logics which are very useful for the representation of ontologies, and, consequently, for the Semantic Web, namely DLs and OWL. We will present several results, both positive and negative, as well as certain heuristics that allow us to determine the AGM-compliance of such logics under both the CVA and the OVA model.

5.2. DLs under the CVA Model

In this section, we will show that the CVA model is incompatible with the AGM theory in all but some trivial DLs. This result is based on the fact that the mere existence of some namespace elements is, by itself, insufficient to imply any non-trivial expression (or set of expressions) that contains them. Intuitively, in the lattice that represents a DL under the CVA model, all existence assertion axioms appear "above" the other axioms of the DL (i.e., near the top, $\varnothing$, node) because of the (%2) postulate. This property causes the nodes that are "immediately below" the existence assertion axioms to be non-decomposable (see, for example, the nodes {A≡B}, {A≡C} and {B≡C} in figure 4.8, page 97).

As an example, consider a DL <L,Cn> and the set (belief) {A≡B} (for A, B concepts of the DL's namespace); {A≡B} implies (but is not implied by) the belief {%A,%B}. If we attempt to contract {%A} from Cn({A≡B}), the result must contain no element of the namespace other than B, or else the postulates of success (if A is also used) and/or inclusion (if any namespace element other than A, B is used) would be violated. The only expressions that can be formed using B alone are certain trivial expressions such as B≡B, B⊑B etc, which are all implied by %B (so they are equivalent to %B by (%2)). Thus, the result should be Cn({%B}). But then the postulate of recovery is violated because Cn({%A,%B})⊂Cn({A≡B}). The situation presented is typical in all DLs under CVA that contain non-trivial expressions. This result is formally proven in the following theorem:

Theorem 5.1. Consider a DL <L,Cn> under the CVA, with the property that there is at least one set K⊆L such that Cn(K)⊃Cn($\cup_{A\in U(K)}$\{%A\}), U(K)≠∅ and U(K) is finite. Then <L,Cn> is not AGM-compliant.

Proof

Consider the following family of sets of axioms:

$\wp$=\{X⊆L | Cn(X)⊃Cn($\cup_{A\in U(X)}$\{%A\}), U(X)≠∅, U(X) is finite\}.

$\wp$≠∅ by hypothesis.

We select a set K∈$\wp$: |U(K)|≤|U(X)| for all X∈$\wp$.

The fact that U(X) is finite for every X∈$\wp$, guarantees the existence of such a K. If there are more than one with this property, we select one arbitrarily.

Set m=|U(K)|>0, K′=$\cup_{A\in U(K)}$\{%A\}⊆L. We will prove that K′⁻(K)=∅.

Initially notice that U(K)≠∅, thus there is some A∈U(K). We conclude that %A∈Cn(K′); but %A∉Cn(∅), because U(%A)⊃U(∅), thus Cn(K′)≠Cn(∅).

Furthermore, by the selection of K we conclude that: Cn(K′)⊂Cn(K).

Summarizing these results: Cn(∅)⊂Cn(K′)⊂Cn(K).

Furthermore: U(K′)=U(K).

Consider some Y⊆L such that Y∈K′⁻(K).

Then: Cn(Y)⊂Cn(K) and Cn(Y∪K′)=Cn(K).

Since Cn(Y)⊂Cn(K), we conclude, by lemma 3.1, that U(Y)⊆U(K)=U(K′), so:

|U(Y)|≤|U(K)|=m.

Suppose that |U(Y)|=0; then U(Y)=∅⊆U(∅) so Cn(Y)=Cn(∅), so Cn(K)=Cn(Y∪K′)=Cn(K′), a contradiction.

Suppose that 0<|U(Y)|<m; then, U(Y)≠∅ and U(Y) is finite; if we assume that Cn(Y)⊃Cn($\cup_{A\in U(Y)}$\{%A\}) then Y∈$\wp$ and |U(Y)|<|U(K)|, a contradiction by the selection of K. Thus, Cn(Y)=Cn($\cup_{A\in U(Y)}$\{%A\}).

Given that U(Y)⊆U(K), we conclude that:

Cn(Y)=Cn($\cup_{A\in U(Y)}$\{%A\})⊆Cn($\cup_{A\in U(K)}$\{%A\})=Cn(K′), thus Cn(Y∪K′)=Cn(K′)⊂Cn(K), a contradiction, since Y∈K′⁻(K).

The only case left is to consider is when |U(Y)|=|U(K)|. But then, given that U(Y)⊆U(K), we conclude that U(Y)=U(K)=U(K′).

Thus Cn(Y)⊇Cn($\cup_{A\in U(Y)}$\{%A\})=Cn($\cup_{A\in U(K)}$\{%A\})=Cn(K′).

Since Y∈K′⁻(K), we get that Cn(Y)⊂Cn(K) and Cn(K)=Cn(K′∪Y)=Cn(Y), a contradiction.

Consequently, there is no set Y∈K′⁻(K), so <L,Cn> is not AGM-compliant. □

Notice that this theorem was proven without any reference to the operators, the connectives or any other property related to the underlying DL. Only a certain condition was assumed to hold, which is true for all but the most trivial DLs:

Corollary 5.1. Any DL under CVA that is at least as expressive as FL_0 and whose namespace allows at least two concepts is not AGM-compliant.

This corollary is immediate by the fact that {A≡B} satisfies the preconditions of theorem 5.1 and can be defined in FL_0. The DL FL_0 is a very poor DL; to the author's knowledge all DLs that have been studied in the literature are more expressive than FL_0; in a sense, FL_0 represents the least expressive DL worth studying, so all DLs actually build on FL_0, adding certain features that are deemed useful by their creators for the application at hand. This fact supports the conclusion that the AGM theory of contraction is not compatible with the CVA

and the existence assertion operator in DLs. Thus, if we wish to use the CVA model for a DL, we need to introduce a different set of rationality postulates to handle contraction.

5.3. OVA Model: Conditions for AGM-Compliance

Introduction

The negative result of the last section forces us to drop the CVA and the existence assertion operator with respect to AGM-compliance and focus our study on the OVA model. For DLs under the OVA model, things are more promising and more interesting: as we will show in this (and the following) section, there are several DLs (under OVA) which are AGM-compliant, as well as several ones which are not. We will split our study to AGM-compliant and non-AGM-compliant DLs, and show several results and heuristics that allow the determination of this property.

Intuitively, we expect at least *some* DLs to be AGM-compliant: there is a known result from the literature that identifies certain DLs (under OVA) as equally expressive to certain logics which belong to the class AGM considered, such as (fragments of) FOL [Bor96], [BCMNPS02]. Given this fact and corollary 4.9, we conclude that these logics are AGM-compliant. But, are there any DLs not satisfying the AGM assumptions which are AGM-compliant? And which are those? These are the main questions that will be dealt with in the following subsections.

Initial Thoughts and a Naïve Approach

Initially, consider a DL <L,Cn> allowing for at least the top concept $\top$ and the connective $\sqsubseteq$ (applicable to concept terms, at least), plus an arbitrary number of other connectives and/or operators. These assumptions will be enough to show most of our results; whenever more assumptions are required, these will be explicitly mentioned in the text.

Our first, and most important, result will be obtained in a constructive manner: we will try to find some member of the set $X^{-}(K)$ for arbitrary K, $X \subseteq L$. Let us start with a (seemingly) simple case: consider two sets of DL axioms, of the form $K=\{A \sqsupseteq \top\}$, $X=\{B \sqsupseteq \top\}$. The symbols A and B may refer to arbitrarily complex concept terms; they need not be namespace elements. Now suppose that the following condition holds: $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$.

We seek a complement of X with respect to K, i.e., a set $Z \subseteq L$ such that $Cn(Z) \subset Cn(K)$ and $Cn(K)=Cn(X \cup Z)$. Let's consider $Z=\{A \sqsupseteq B\}$; this set is a good candidate for the required Z, because it is obvious that $Cn(Z) \subseteq Cn(K)$ and $Cn(X \cup Z)=Cn(K)$. There is a catch though: Z is not always a proper implication of K, as required by definition 4.2; sometimes it so happens that $Cn(Z)=Cn(K)$:

Example 5.1. (Example provided by Thomas Studer on a personal communication).
Consider the DL ALC with concept hierarchies, which allows the operators $\neg$, $\sqcap$, $\sqcup$, $\forall$, $\exists$, $\bot$, $\top$ and the connectives $\equiv$ and $\sqsubseteq$, applicable to concept terms only. Consider also some concept A and some role R. Set:
$K=\{A \sqsupseteq \top\}$, $X=\{\neg A \sqcup \exists R.A \sqcup \forall R.\bot \sqsupseteq \top\}$.
We define Z as above, i.e., $Z=\{A \sqsupseteq \neg A \sqcup \exists R.A \sqcup \forall R.\bot\}$.
We will show that $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$, but $Cn(Z)=Cn(K)$, so $Z \notin X^{-}(K)$.
Initially, consider the interpretation $(\Delta^I,\cdot^I)$ such that $\Delta^I=\{a,b\}$, $A^I=\{a\}$, $R^I=\{(a,b)\}$.
It can be shown that for this interpretation: $(\neg A \sqcup \exists R.A \sqcup \forall R.\bot)^I=\{b\} \neq \Delta^I$.
We found an interpretation for which X is not satisfied, so $Cn(X) \neq Cn(\varnothing)$.

On the other hand, if you take any arbitrary interpretation $(\Delta^I,\cdot^I)$ which satisfies K, then $A^I=\Delta^I$. Take any $x\in\Delta^I$ in such an interpretation. If there is a $y\in\Delta^I$ such that $(x,y)\in R^I$, then $x\in(\exists R.A)^I$; if, on the other hand, there is no such y, then $x\in(\forall R.\bot)^I$. Therefore, in either case: $x\in(\neg A\sqcup\exists R.A\sqcup\forall R.\bot)^I$ for all $x\in\Delta^I$, so X is satisfied. We conclude that X is satisfied whenever K is satisfied, so $K\vdash X$, i.e., $Cn(X)\subseteq Cn(K)$.
Finally, take an interpretation $(\Delta^I,\cdot^I)$ such that $A^I=\varnothing$, $R^I=\varnothing$. Then K is obviously not satisfied, but X is, because $(\forall R.\bot)^I=\Delta^I$. Thus $X\nvdash K$.
Combining these three facts, we get: $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$, so the selected K, $X\subseteq L$ have the desired properties.
We will now show that Cn(Z)=Cn(K).
Using well known results from the DL literature:
$A\sqsupseteq\neg A\sqcup\exists R.A\sqcup\forall R.\bot\Leftrightarrow$
$A\sqcup\neg(\neg A\sqcup\exists R.A\sqcup\forall R.\bot)\equiv\top\Leftrightarrow$
$A\sqcup(A\sqcap\forall R.\neg A\sqcap\exists R.\top)\equiv\top\Leftrightarrow$
$A\sqsupseteq\top$
The latter equivalence is true because $(A\sqcap\forall R.\neg A\sqcap\exists R.\top)^I\subseteq A^I$ for any interpretation $(\Delta^I,\cdot^I)$, thus $(A\sqcup(A\sqcap\forall R.\neg A\sqcap\exists R.\top))^I=A^I$ for any interpretation $(\Delta^I,\cdot^I)$.
We conclude that Cn(K)=Cn(Z); so, for this particular example, $Z\notin X^-(K)$.

The Need for Transformations

Example 5.1 shows that we are close to finding the required Z, but not yet there. Our initial (and naïve) idea must be somehow refined to guarantee that $Cn(Z)\neq Cn(K)$ in all cases. The following lemma will prove helpful:
Lemma 5.1. Consider a DL <L,Cn>, two arbitrarily complex concept terms A, B and the sets $K=\{A\sqsupseteq\top\}$, $X=\{B\sqsupseteq\top\}$. If there is some interpretation $(\Delta^I,\cdot^I)$ for which $B^I=\varnothing$, then $X^-(K)\neq\varnothing$.
Proof
By lemma 4.6, we only need to check the case where $Cn(\varnothing)\subset Cn(X)\subset Cn(K)$.
Consider, as in the naïve case, the set $Z=\{A\sqsupseteq B\}$.
We will show that $Z\in X^-(K)$.
Indeed:
$Cn(Z\cup X)=Cn(K)$ (because, by the transitivity of $\sqsupseteq$, $\{A\sqsupseteq B, B\sqsupseteq\top\}\vdash\{A\sqsupseteq\top\}$ and by the fact that $K\vdash X$, $K\vdash Z$, we get $K\vdash Z\cup X$, so $K\cong Z\cup X$).
$Cn(Z)\subseteq Cn(K)$ (because whenever $A^I=\Delta^I$ it holds that $A^I\supseteq B^I$, so, whenever K is satisfied, Z is satisfied).
It remains to show that $Cn(Z)\neq Cn(K)$. By the hypothesis, there is some interpretation $(\Delta^I,\cdot^I)$ for which $B^I=\varnothing$. Obviously, X is not satisfied by this particular interpretation, so K is not satisfied either (because $K\vdash X$). On the other hand, Z is satisfied by this interpretation (because $A^I\supseteq\varnothing=B^I$).
So, we have found an interpretation satisfying Z but not K. We conclude that $Cn(Z)\neq Cn(K)$.
Summing up these results: $Cn(Z)\subset Cn(K)$, $Cn(Z\cup X)=Cn(K)$, so $Z\in X^-(K)$ and the proof is complete. □

This lemma shows that the addition of an extra assumption on X gives us the desired result. Of course, lemma 5.1 is not enough to show a DL to be decomposable: our definition requires that $X^-(K)\neq\varnothing$ for all pairs K, $X\subseteq L$ in order for a logic to be decomposable, not just for pairs of a particular form. Despite this deficiency, lemma 5.1 forms the backbone of our main theorem regarding the AGM-compliance of DLs.

To get to a more generalized version of lemma 5.1, we will have to progressively relax our assumptions on K, X in order to reach the desired generality; this relaxation of assumptions from K, X may force us to impose certain additional assumptions on the logic <L,Cn> instead. Our first generalization shows that the result of lemma 5.1 can be easily generalized to the case when K is not a singular set:

Lemma 5.2. Consider a DL <L,Cn> and the sets of axioms $K=\{A_j \sqsupseteq \top \mid j \in J\}$ and $X=\{B \sqsupseteq \top\}$. If there is an interpretation $(\Delta^I,\cdot^I)$ such that $B^I=\emptyset$, then $X^-(K)\neq\emptyset$.

Proof

The result can be shown by repeating the steps of the proof of lemma 5.1 for the set: $Z=\{A_j \sqsupseteq B \mid j \in J\}$. □

Our second (and final) generalization is based on lemma 4.2 and lemma 4.3:

Theorem 5.2. Consider a DL <L,Cn> admitting (at least) the connective $\sqsubseteq$ (applicable for concept terms, at least) and the constant operator $\top$. Suppose also that we can define two functions T_1, T_2 with the following properties:

- $T_1:P(L)\rightarrow P(L)$, $T_2:P(L)\rightarrow P(L)$
- For all $Y\subseteq L$, $T_1(Y)=\{A_j \sqsupseteq \top \mid j \in J\}$ for some index set J and concept terms A_j
- For all $Y\subseteq L$, $T_1(Y)\cong Y$
- For all $Y\subseteq L$, $T_2(Y)=\{B \sqsupseteq \top\}$ for some concept term B for which there is some interpretation $(\Delta^I,\cdot^I)$ such that $B^I=\emptyset$
- For all $Y\subseteq L$, $Cn(Y)\neq Cn(\emptyset)$ implies that $Cn(\emptyset)\subset Cn(T_2(Y))\subseteq Cn(Y)$

Then, <L,Cn> is AGM-compliant.

Proof

Consider any two sets $K, X\subseteq L$ such that $Cn(\emptyset)\subset Cn(X)\subset Cn(K)$.

Set $K'=T_1(K)$, $X'=T_2(X)$.

By the assumptions on T_1, T_2 we conclude that K', X' are of the form required by lemma 5.2, so $X'^-(K')\neq\emptyset$.

By lemma 4.2 and the fact that $T_1(K)\cong K$ we get: $X'^-(K')=X'^-(K)$.

By lemma 4.3 and the fact that $Cn(\emptyset)\subset Cn(T_2(X))\subseteq Cn(X)\subset Cn(K)$ we get: $X'^-(K)\subseteq X^-(K)$.

Therefore: $X^-(K)\supseteq X'^-(K)=X'^-(K')\supset\emptyset$.

We conclude that <L,Cn> is decomposable, thus AGM-compliant. □

Theorem 5.2 provides a way to show a certain DL AGM-compliant, but it is based on the important underlying assumption that the two functions (transformations) T_1 and T_2 can be defined in the DL under question. So, if we can define these functions, we can show AGM-compliance; this reduces the problem of determining AGM-compliance to the problem of finding two transformations with the properties described in theorem 5.2. The question that remains to be resolved is, in which DLs do these functions (transformations) exist?

Defining the Transformations

Table 5.1 provides the basis for both these transformations; this table shows how each of the commonly used types of DL axioms can be equivalently written into the form "$A \sqsupseteq \top$" for some (possibly complex) concept term A. For example, the axiom {disj(A,B)} is equivalent to the axiom $\{\neg A \sqcup \neg B \sqsupseteq \top\}$, for any concept terms A, B. These transformations do not involve concept terms only; a glance at the table shows that axioms involving individuals or roles also have an equivalent counterpart of the form "$A \sqsupseteq \top$". All these equivalences can be shown relatively easily using model-theoretic arguments and the DL semantics; many of them

Axiom	Equivalent axiom of the proper form	Required operators
$A \sqsubseteq B$	$\neg A \sqcup B \sqsupseteq \top$	$\neg, \sqcup$
$R \sqsubseteq S$	$\forall(R \sqcap \neg S).\bot \sqsupseteq \top$	$\bot, \forall, \neg_R, \sqcap_R$
$A \not\sqsubseteq B$	$\exists \top_R.(A \sqcap \neg B) \sqsupseteq \top$	$\neg, \sqcap, \exists, \top_R$
$R \not\sqsubseteq S$	$\exists \top_R.\exists(R \sqcap \neg S).\top \sqsupseteq \top$	$\exists, \top_R, \neg_R, \sqcap_R$
$A \equiv B$	$(\neg A \sqcup B) \sqcap (A \sqcup \neg B) \sqsupseteq \top$	$\neg, \sqcup, \sqcap$
$R \equiv S$	$\forall(R \sqcap \neg S).\bot \sqcap \forall(S \sqcap \neg R).\bot \sqsupseteq \top$	$\bot, \sqcap, \forall, \neg_R, \sqcap_R$
$A \not\equiv B$	$\exists \top_R.[(A \sqcap \neg B) \sqcup (B \sqcap \neg A)] \sqsupseteq \top$	$\neg, \sqcap, \sqcup, \exists, \top_R$
$R \not\equiv S$	$\exists \top_R.\exists[(\neg R \sqcap S) \sqcup (\neg S \sqcap R)].\top \sqsupseteq \top$	$\exists, \top_R, \neg_R, \sqcap_R, \sqcup_R$
$A \sqsubset B$	$(\neg A \sqcup B) \sqcap \exists \top_R.(B \sqcap \neg A) \sqsupseteq \top$	$\neg, \sqcap, \sqcup, \exists, \top_R$
$R \sqsubset S$	$\forall(R \sqcap \neg S).\bot \sqcap \exists \top_R.\exists(S \sqcap \neg R).\top \sqsupseteq \top$	$\bot, \sqcap, \exists, \forall, \top_R, \neg_R, \sqcap_R$
$A / \sqsubset B$	$\forall \top_R.\exists \top_R.(A \sqcap \neg B) \sqcup \forall \top_R.(\neg B \sqcup A) \sqsupseteq \top$	$\neg, \sqcap, \sqcup, \exists, \forall, \top_R$
$R / \sqsubset S$	$\forall \top_R.\exists \top_R.\exists(R \sqcap \neg S).\top \sqcup$ $\forall \top_R.\forall(S \sqcap \neg R).\bot \sqsupseteq \top$	$\bot, \sqcup, \exists, \forall, \top_R, \neg_R, \sqcap_R$
disj(A,B)	$\neg A \sqcup \neg B \sqsupseteq \top$	$\neg, \sqcup$
disj(R,S)	$\forall(R \sqcap S).\bot \sqsupseteq \top$	$\bot, \forall, \sqcap_R$
A(a)	$\neg\{a\} \sqcup A \sqsupseteq \top$	$\neg, \sqcup, \{\ldots\}$
R(a,b)	$\exists R.\{b\} \sqcup \neg\{a\} \sqsupseteq \top$	$\neg, \sqcup, \exists, \{\ldots\}$
a=b	$\neg\{a\} \sqcup \{b\} \sqsupseteq \top$	$\neg, \sqcup, \{\ldots\}$
a≠b	$\neg\{a\} \sqcup \neg\{b\} \sqsupseteq \top$	$\neg, \sqcup, \{\ldots\}$

Table 5.1. Transforming DL axioms into the form $A \sqsupseteq \top$

are based on well-known results from the literature. In table 5.1, A, B refer to concept terms, R, S refer to role terms and a, b refer to individuals. All operators subscripted by $\cdot_R$ (in the third column) apply to role terms; the $\cdot_R$ subscript has been dropped in the second column for readability purposes, as the range of the operator should be clear from the context. The other operators apply to concept terms or individuals, depending on the context. The range of each connective (concepts, roles or individuals) should also be clear from the context. The symbol $\top_R$ refers to the top role, i.e., the role connecting every individual to every individual and the connective $/\sqsubset$ stands for non-proper-inclusion (i.e., the complement of $\sqsubset$). The symbols $\neg$ and $\exists$ refer to full (rather than atomic) negation and full (rather than limited) existential quantification respectively.

Given table 5.1, both T_1 and T_2 can be easily defined. Indeed, take any set $Y \subseteq L$. For T_1, all we have to do is to take each axiom $y \in Y$ and transform it according to table 5.1 to its "$A \sqsupseteq \top$" equivalent counterpart; this gives us a set Y' of the form $\{A_j \sqsupseteq \top \mid j \in J\}$ which is (obviously) equivalent to Y.

For T_2 things are a little more complicated. If $Cn(Y)=Cn(\varnothing)$ then set $T_2(Y)=\{\bot \sqsupseteq \top\}$; this set is contradictory and totally unrelated to Y, however, the phrasing of theorem 5.2 does not disallow this. For the more interesting case when $Cn(Y) \neq Cn(\varnothing)$, we must first select one non-

tautological axiom from Y, say $y_0 \in Y$ (such an axiom obviously exists, otherwise $Cn(Y)=Cn(\varnothing)$). Then we must apply the relevant transformation from table 5.1 to y_0. Suppose that the resulting axiom is $B_0 \sqsupseteq \top$. Set $T_2(Y)=\{\forall \top_R.B_0 \sqsupseteq \top\}$. The following remarks show that the defined T_2 satisfies the properties required by theorem 5.2:

- $T_2(Y)$ is of the required form ($\{\forall \top_R.B_0 \sqsupseteq \top\}$, i.e., $\{B \sqsupseteq \top\}$).
- Suppose that for all interpretations $(\Delta^I,\cdot^I)$, $B_0^I=\Delta^I$. Then the axiom $B_0 \sqsupseteq \top$ is tautological, so y_0 is also tautological (a contradiction by the selection of y_0); thus, there is some interpretation for which $B_0^I \neq \Delta^I$; for this very interpretation it holds that $(\forall \top_R.B_0)^I=\varnothing$ (can be easily proven using model theoretic arguments and the semantics of $\forall$ and $\top_R$); thus, there is some interpretation $(\Delta^I,\cdot^I)$ for which $(\forall \top_R.B_0)^I=\varnothing$.
- Since $y_0 \in Y$, we conclude that $Y \vdash \{y_0\} \cong \{B_0 \sqsupseteq \top\}$.
- For all interpretations $(\Delta^I,\cdot^I)$ and all concept terms B_0, the relation $B_0^I=\Delta^I$ implies that $(\forall \top_R.B_0)^I=\Delta^I$ (can be easily proven using model theoretic arguments and the semantics of $\forall$ and $\top_R$); thus, $\{B_0 \sqsupseteq \top\} \vdash \{\forall \top_R.B_0 \sqsupseteq \top\}$, so, ultimately, $Y \vdash \{y_0\} \vdash \{\forall \top_R.B_0 \sqsupseteq \top\} \vdash T_2(Y)$.
- For the interpretation constructed in the second bullet, $T_2(Y)$ is not satisfied, so $T_2(Y)$ is not tautological.

Combining all these remarks, we conclude that both T_1 and T_2 satisfy the properties required by theorem 5.2. Therefore, the equivalences of table 5.1 provide the necessary tools for the definition of both transformations.

Applications and Additional Results

One problem with the transformations T_1 and T_2 is that they are not definable in all DLs. For example, if the DL under question does not allow the set (one-of) operator ($\{\ldots\}$), then the axiom $\neg\{a\} \sqcup A \sqsupseteq \top$ is not expressible, so the equivalence relation $\{A(a)\} \cong \{\neg\{a\} \sqcup A \sqsupseteq \top\}$ cannot be applied in this DL; thus, we cannot define the necessary transformations and cannot transform the axiom A(a) into the required form. Therefore, theorem 5.2 and the analysis that followed it do not show that all DLs under the OVA model are AGM-compliant; they show that, if the transformations are definable for some DL, then this DL is AGM-compliant (the third column of table 5.1 shows the operators that are required in the DL for each transformation to be definable). Thus, we don't have any direct answer on the AGM-compliance of any given DL yet; we need an additional step that will determine the DLs in which the transformations are definable. As we will see, this fact makes theorem 5.2 extremely general, forming the basis for a variety of positive results regarding the AGM-compliance of DLs.

The most straightforward application of theorem 5.2 uses all the connectives and operators that appear in table 5.1, the operator $\top$ and the connective $\sqsubseteq$ (which are necessary to form the expression $A \sqsupseteq \top$ for the basic case), as well as all the operators and connectives that are necessary for the definition of T_1 and T_2, given the equivalences of table 5.1. Gathering all these requirements we get the following (rather naïve) initial corollary:

Corollary 5.2. A DL containing the constants $\top$, $\bot$, $\top_R$, the operators $\neg$, $\sqcap$, $\sqcup$, $\exists$, $\forall$, $\neg_R$, $\sqcap_R$, $\sqcup_R$, $\{\ldots\}$, the concept connective $\sqsubseteq$ plus any of the connectives that appear in table 5.1 is AGM-compliant.

Corollary 5.2 describes one possible set of prerequisites for the definition of one possible pair of transformations, in effect providing one possible application of theorem 5.2. However, the family of DLs described by this corollary is not the only AGM-compliant one. Several refinements, improvements and variations of this corollary can be formulated. A simple check

of the operators required reveals that certain operators (like $\sqcup$, $\exists$, $\sqcup_R$, $\bot$, $\top$, $\top_R$) are easily expressible in terms of other operators using certain well-known equivalences from the literature, namely:

$A \sqcup B \equiv \neg(\neg A \sqcap \neg B)$

$\exists R.A \equiv \neg \forall R.\neg A$

$R \sqcup S \equiv \neg(\neg R \sqcap \neg S)$

$\bot \equiv \neg A \sqcap A$, for some concept A ($A \in N_{DL}^{C}$)

$\top \equiv \neg \bot \equiv \neg(\neg A \sqcap A)$, for some concept A ($A \in N_{DL}^{C}$)

$\top_R \equiv \neg R \sqcup R \equiv \neg(R \sqcap \neg R)$, for some role R ($R \in N_{DL}^{R}$)

This analysis shows that these operators can be removed from our requirements for an AGM-compliant logic in corollary 5.2: if the operator $\sqcup$ is necessary for some transformation, then it can be replaced using $\neg$, $\sqcap$ (using the aforementioned equivalences) to create an alternative (and equivalent) transformation which does not need it. The same holds for the other redundant operators as well. Also, notice that several operators are interchangeable: for example any of $\exists$, $\forall$ could be used, but we don't need both; any of the pairs $\neg$, $\sqcap$ or $\neg$, $\sqcup$ could be used, but we don't need both; and so on. For the constants ($\top$, $\bot$, $\top_R$), the existence of at least one concept or role (respectively) in the namespace is required. This is an additional assumption, but it's so trivial that it's not even worth mentioning. The concept/role used need not be a "fresh" one, i.e., it may appear in the DL KB elsewhere, so all we require is that N_{DL}^{C} and N_{DL}^{R} are non-empty. This analysis leads to several similar corollaries which are easy to get. Here is one possibility:

Corollary 5.3. A DL containing the operators $\neg$, $\sqcap$, $\forall$, $\neg_R$, $\sqcap_R$, $\{\ldots\}$, the concept connective $\sqsubseteq$ plus any of the connectives that appear in table 5.1 and has the property that $N_{DL}^{C} \neq \varnothing$, $N_{DL}^{R} \neq \varnothing$ is AGM-compliant.

Perhaps the most important generalization of our initial results occurs by the addition of more operators to an AGM-compliant logic. If a certain transformation is definable in a DL, then, obviously, the addition of a new operator in the logic does not change this fact, even if the new operator increases the expressive power of the DL at hand. Thus, corollary 5.2 and corollary 5.3 provide a *minimal* set of operators required to guarantee AGM-compliance. Any additional operators do not bar AGM-compliance, because the transformations can still be defined. Thus, all logics that contain more operators than the DLs described in these corollaries are AGM-compliant too.

For the connectives, things are the exact opposite: adding connectives that are not considered by table 5.1 might cause problems, because the addition of connectives results to new axiom types. These new axiom types may not be transformable to the desired form of $A \sqsupseteq \top$, or they may be transformable, but require the use of new operators and transformations for this purpose, which are not considered in table 5.1.

This observation can be applied in the opposite direction as well: the operators we need to guarantee AGM-compliance are just those that are required to produce the necessary transformations, i.e., exactly those that are required to transform each allowed axiom type to the form $A \sqsupseteq \top$. If some of the axiom types of table 5.1 are not allowed in the underlying DL, then some of the equivalences of table 5.1 become irrelevant and need not be supported through the use of operators. For example, if we are interested in DL KBs without an Abox, then the axiom types containing individuals are not interesting for our purposes, because they don't appear in our DL. Thus, the equivalences related to such axioms are irrelevant as far as

the transformations are concerned. Since the one-of operator {…} is only used for the transformation of these axiom types, once these axiom types are removed, this operator can also be removed from the minimal set of operators required for AGM-compliance. Similarly, certain logics disallow certain connectives or certain uses of ones. Such restrictions might affect (i.e., reduce) the required minimal operator set (by allowing simpler transformations or making certain equivalences irrelevant). A full account of such cases is outside the scope of this discussion.

Notice however, that this observation is true as long as the concept connective $\sqsubseteq$ remains in the logic at hand; in a different case, the results of this subsection are not applicable, because in the phrasing of theorem 5.2 (as well as in corollary 5.2 and corollary 5.3) the use of the operator $\sqsubseteq$ (for concept terms) is mandatory. However, using the concept of AGM-equivalence, we can show that this is an unnecessary restriction:

Lemma 5.3. Consider two DLs <L,Cn>, <L′,Cn′> with the following properties:

- The two DLs have identical namespaces (i.e., $N_{DL}=N_{DL}'$)
- The two DLs contain the same operators (i.e., $O_{DL}=O_{DL}'$)
- The two DLs contain at least the operators $\top$, $\neg$, $\forall$, $\top_R$ (i.e., $O_{DL}=O_{DL}'\supseteq\{\top, \neg, \forall, \top_R\}$)
- All the connectives of any of the two DLs appear in table 5.1
- The DL <L,Cn> contains at least one of the concept connectives of table 5.1
- The DL <L′,Cn′> contains exactly the concept connective $\sqsubseteq$ and all the connectives of <L,Cn> (i.e., $C_{DL}\cup\{\sqsubseteq\}=C_{DL}'$)

Then the two DLs are AGM-equivalent.

Proof

If $\sqsubseteq\in C_{DL}$ then the two DLs are identical, so the proof is trivial.

Suppose that $\sqsubseteq\notin C_{DL}$.

Obviously $L\subseteq L'$.

The consequences of a set $X\subseteq L\subseteq L'$ do not change if we view it under L′; therefore:

$Cn(X)=Cn'(X)\cap L$.

Finally, the following equivalences can be easily shown using model-theoretic arguments:

$\{A\sqsupseteq\top\} \Leftrightarrow \{A\equiv\top\} \Leftrightarrow \{\neg\forall\top_R.A\sqsubset\top\} \Leftrightarrow \{\neg\forall\top_R.A\not\equiv\top\} \Leftrightarrow \{\neg\forall\top_R.A\not\sqsupseteq\top\} \Leftrightarrow \{A/\sqsubset\top\} \Leftrightarrow \{disj(\top,\neg A)\}$

These equivalences use operators that are admitted in both DLs and allow us to transform any axiom that uses $\sqsubseteq$ into an axiom that does not contain $\sqsubseteq$ and is admitted in the DL <L,Cn>. Notice that the axioms containing $\sqsubseteq$ are exactly the axioms that are contained in L′\L and that these equivalences do not change the consequences of $A\sqsupseteq\top$, in terms of the logic <L′,Cn′>.

More formally, these equivalences, allow us to define a function $f:L'\backslash L\rightarrow P(L)$ for which $Cn'(\{x\})=Cn(f(x))$ for all $x\in L'\backslash L$.

Combining these facts with proposition 4.4 gives the desired result. □

In short, lemma 5.3 uses the equivalences: $\{A\sqsupseteq\top\} \Leftrightarrow \{A\equiv\top\} \Leftrightarrow \{\neg\forall\top_R.A\sqsubset\top\} \Leftrightarrow \{\neg\forall\top_R.A\not\equiv\top\} \Leftrightarrow \{\neg\forall\top_R.A\not\sqsupseteq\top\} \Leftrightarrow \{A/\sqsubset\top\} \Leftrightarrow \{disj(\top,\neg A)\}$ to show that any DL that contains any of the usual concept connectives is equivalent to a DL that contains the same connectives, plus the inclusion concept connective ($\sqsubseteq$), provided that the operators necessary to define these identities (i.e., $\top$, $\neg$, $\forall$, $\top_R$) are allowed in both DLs. Therefore, we can use the previous results to show that the DL that contains $\sqsubseteq$ is AGM-compliant and then propagate the result to the one that does not contain $\sqsubseteq$ using AGM-equivalence. This allows us to generalize the previous corollaries (corollary 5.2 and corollary 5.3) to logics that do not

contain $\sqsubseteq$, under the conditions of lemma 5.3. However, it should be noticed that lemma 5.3 fails if no operator applicable to concepts is allowed in the underlying DL, or if the required operators are not allowed.

But again, these are not the only positive results related to AGM-compliance that can be derived from theorem 5.2; more alternatives may be possible. Notice that theorem 5.2 requires that two transformations with the desired properties exist, but does not restrict us to use the transformations that we proposed. In effect, one could replace the equivalences of table 5.1 with another set of equivalences; this would lead to different transformations, which could, possibly, have different requirements regarding the necessary operators. Similarly, the transformations themselves could be defined differently, with or without the equivalences of table 5.1. All these variations could lead to different minimal sets of operators required to guarantee AGM-compliance.

One example of such an alternative is to change transformation T_2. In the alternative transformation, we replace the constant $\top_R$ with a "fresh" role name from the DL namespace (i.e., a role name that does not appear in the input). This way, if the result of T_2 upon a set $Y \subseteq L$ is $\{\forall \top_R.B_0 \sqsupseteq \top\}$, then the alternative transformation (say T_2') would give $\{\forall R.B_0 \sqsupseteq \top\}$ for a role name R that does not appear in B_0. This way, we drop the requirement to introduce the constant $\top_R$, but introduce a new requirement, i.e., that the DL namespace must contain an infinite number of role names (so we can always find one that is "fresh").

The validity of this alternative transformation T_2' (i.e., that it satisfies the properties required by theorem 5.2) can be showed by following similar steps as in the proof of the validity of the original transformation (T_2). This proof is based on the facts:

- $\{B_0 \sqsupseteq \top\} \vdash \{\forall R.B_0 \sqsupseteq \top\}$ for all roles R and concept terms B_0.
- The interpretation of R (R^I) is independent of the interpretation of B_0 (B_0^I), because R is a fresh role. Therefore, we can restrict ourselves to the interpretations for which $R^I=\Delta^I \times \Delta^I$ to show that $Cn(T_2(Y)) \neq Cn(\varnothing)$ and that there is some interpretation for which $(\forall R.B_0)^I=\varnothing$ using identical steps as in the proof of the validity of T_2.

These two observations allow us to make the relevant proof. The existence of this alternative transformation shows that there are certain AGM-compliant DLs which are not covered by the other results of this subsection.

A further source of generality is the fact that in some DLs we may be able to greatly simplify the definition of the more complex transformation, i.e., T_2. For most logics and most concept terms (say B) of the logic, there is some interpretation for which $B^I=\varnothing$. If this is the case, the final step of the transformation T_2 (and T_2') where we add the clause "$\forall \top_R$." (or "$\forall R$." respectively) is no longer necessary; this last step is crucial to guarantee that the fourth bullet in the phrasing of theorem 5.2 is satisfied. If this bullet is satisfied anyway, we need not add any more complexity to our expression, so we can keep the result of the previous step of the transformation ($B_0 \sqsupseteq \top$) and present it to the output of T_2 (or T_2').

This analysis shows that there are several possible applications of theorem 5.2. Unfortunately, there is no single operator and/or connective that guarantees AGM-compliance in a DL; rather, AGM-compliance derives from adequate blends of operators and connectives which allow the definition of the transformations of theorem 5.2. This makes the analysis, and, consequently, the identification of AGM-compliant DLs, rather complicated, as AGM-compliance should be determined on a per-case basis. The aim of the discussion that appeared in this subsection was not to exhaustively list all the different possibilities; rather, we aimed at

showing the diversity of the different results that can be proven using theorem 5.2. We believe that a careful study of DLs and their semantics may lead to further, yet unexplored possibilities regarding the transformations that would show more DLs to be AGM-compliant.

Discussion

Corollary 5.3 (and similar results) require certain operators in order to guarantee AGM-compliance. Most of these operators (like $\neg$, $\sqcap$, $\forall$) are standard in most interesting DLs. One exception is the operator $\{\ldots\}$, which is common in many DLs, but could not be classified as "standard". Fortunately, as explained before, this operator is not necessary for AGM-compliance if we assume an empty Abox in the DL under question. If $\{\ldots\}$ is allowed, then our results apply also to DL KBs that contain a non-empty Abox.

A more important problem in this respect is posed by the role operators ($\neg_R$, $\sqcap_R$), which do not appear in most DLs. A glance at table 5.1 shows that these operators are necessary when role connectives are admitted in the DL at hand; they are also necessary for the original T_2 transformation, because of the "$\forall \top_R$." clause. They are not necessary if no connectives involving roles are allowed in the underlying DL and we use the alternative transformation with the fresh role name (T_2'), or if the addition of the "$\forall \top_R$." clause is not necessary (see the previous subsection). In a later section, we will show that the existence of operators and connectives that involve roles is a very important determinant for the AGM-compliance of the DL under question.

One last (but certainly not least) observation that can be made is that theorem 5.2 and its various corollaries do not provide a complete characterization of AGM-compliance for DLs. However, it looks like this characterization is close to being complete: all the AGM-compliant DLs that we have considered in this thesis fall into one of the theorem's innumerable variations and corollaries; those who don't, eventually turn out to be non-AGM-compliant (see the next sections for some examples). It is part of our future work to determine whether this pattern is simply coincidental or not.

An AGM-compliant Contraction Operator

Of course, the primary result of this section is the proof that certain DLs are AGM-compliant; however, corollary 4.1 implies a secondary effect of our analysis. Using the proofs of lemma 5.1, lemma 5.2 and theorem 5.2 and the result of corollary 4.1, we can get an AGM-compliant contraction operator for use in AGM-compliant DLs.

To define this operator, consider any two sets of axioms $K, X \subseteq L$ in some AGM-compliant DL $\langle L, Cn \rangle$; to define $K - X$, we will distinguish the different cases regarding K and X:

- In the limit cases where $Cn(X)=Cn(\varnothing)$, $Cn(K)=Cn(\varnothing)$ or $Cn(X) \not\subseteq Cn(K)$, we set $K - X = Cn(K)$; the result follows from corollary 4.1, since $K \in X^{-}(K)$ by definition.
- In the case where $Cn(\varnothing) \subset Cn(X) \subseteq Cn(K)$ we apply the transformations T_1 and T_2 (or T_2') upon K and X respectively and then form the set Z, as in the proof of lemma 5.2; this set Z belongs to $X^{-}(K)$ (see the proof of lemma 5.2) and the result follows by corollary 4.1.

The contraction operator thus defined is AGM-compliant (by corollary 4.1). It is also very efficient: performing the transformations (in the second, interesting case) takes only linear time on the number of axioms in the sets K, X, while the calculation in the limit cases only needs constant time. Therefore, the bottleneck of the whole computation is the determination of whether $Cn(\varnothing) \subset Cn(X) \subseteq Cn(K)$ holds. The complexity of this problem depends on the complexity of the underlying DL. However, if we are given an oracle that can determine in

constant time whether a given implication holds or not, then we can compute the result of the contraction operation in linear worst-case complexity. Of course, most interesting DLs have a reasoning complexity much worse than linear (see [BS01], [BCMNPS02], [Zol06]), so the overhead posed by our contraction algorithm is negligible.

Our contraction operator has been formulated in syntactical terms. Its semantical properties have not been studied, because the focus of this thesis is not the development of an explicit ontology contraction algorithm, but the verification that the generalized AGM theory can be used in certain DLs as a postulation method for ontology contraction. Our future work includes the development of a contraction operator which will be based on semantical rather than syntactical considerations, in addition to being AGM-compliant, as well as the study of the semantical properties of the proposed operator.

5.4. OVA Model: Conditions for Non-AGM-Compliance

Results

In the previous section we showed several DLs to be AGM-compliant; unfortunately there are also many DLs which are not. As we will see later, the key reason for this phenomenon is the presence of role axioms in the DL, combined with the disallowance of role connectives (especially $\neg_R$, $\sqcap_R$). The results of this section will be based on the following simple lemma, which is applicable to any logic (not just in DLs):

Lemma 5.4. Consider a logic <L,Cn> and a set $K \subseteq L$. Set $X=\{x \in L \mid Cn(\{x\}) \subset Cn(K)\}$. If $Cn(\varnothing) \subset Cn(X) \subset Cn(K)$ then K is not decomposable (so <L,Cn> is not AGM-compliant).

Proof

Consider the family of sets which contains only X, say $\wp=\{X\}$. Then, the following hold:

- For all $Y \in \wp$, $Cn(Y) \subset Cn(K)$
- For all $Z \subseteq L$ such that $Cn(Z) \subset Cn(K)$ it holds that $Cn(Z) \subseteq X$, so there is some $Y \in \wp$ (namely Y=X) for which $Cn(Z) \subseteq Cn(Y)$
- Obviously Cn(X)=X, so for all $Y \in \wp$, Cn(Y)=Y
- Since $\wp=\{X\}$, a singular set, for all Y, $Y' \in \wp$, $Cn(Y) \subseteq Cn(Y')$ implies $Y=Y'$

These facts imply that $\wp$ is a max-cut of K (see figure 5.1). Moreover:

$Cn(\cap_{Y \in \wp} Cn(Y))=Cn(Cn(X))=Cn(X) \neq Cn(\varnothing)$, and the result follows from theorem 4.3. □

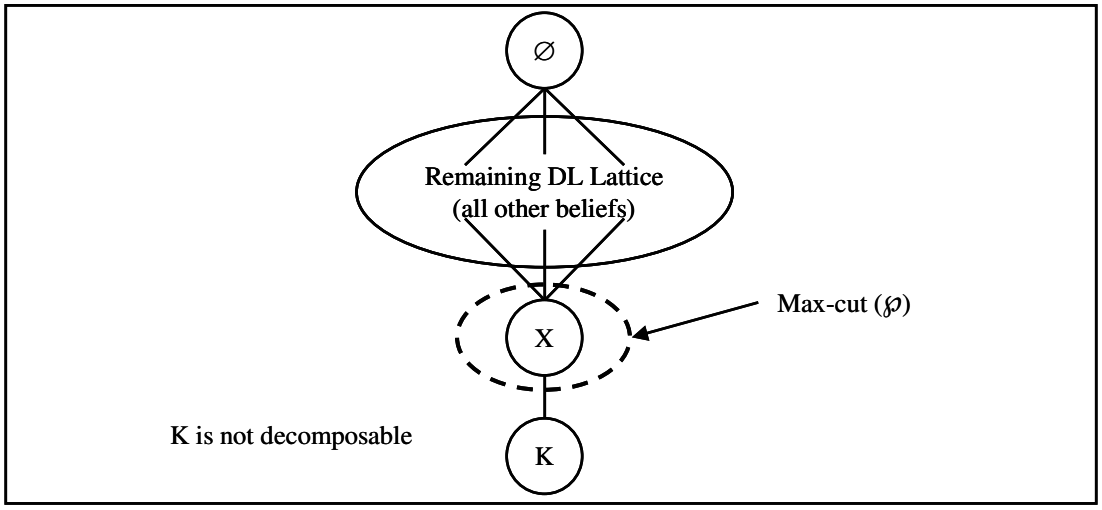

Figure 5.1. The intuition behind lemma 5.4

The intuition behind lemma 5.4 is that, if a logic contains a belief (say K) which cannot be deduced by all its proper consequences combined (say X), then this logic is not AGM-compliant, because $X^{-}(K)=\varnothing$. The situation is shown in figure 5.1.

This lemma is very important for our study because the pattern described by it appears in several DLs. More specifically, this pattern appears in many DLs that admit role axioms of the form $R \sqsubseteq S$ or $R \equiv S$ but do not admit role operators (namely $\neg_R$, $\sqcap_R$); by lemma 5.4, such DLs are not AGM-compliant. The formal phrasing of these facts can be found in the following theorems:

Theorem 5.3. Consider a DL <L,Cn> with the following properties:

- The DL admits at least two role names (say R, S) and one concept name (say A), i.e., $N_{DL}^{R} \supseteq \{R, S\}$, $N_{DL}^{C} \supseteq \{A\}$
- The DL admits at least one of the operators $\forall$, $\exists$ (full or limited), $(\geq_n)$, $(\leq_n)$, for at least some n, i.e., $O_{DL} \cap (\{\forall, \exists\} \cup \{\geq_n, \leq_n \mid n \in \mathbb{N}\}) \neq \varnothing$
- The DL admits any (or none) of the operators $\neg$ (full or atomic), $\sqcap$, $\sqcup$, $^{-}$, $\top$, $\bot$, $\{\ldots\}$, in addition to any of the operators of the previous bullet, i.e., $O_{DL} \subseteq \{\neg, \sqcap, \sqcup, ^{-}, \top, \bot, \{\ldots\}, \forall, \exists\} \cup \{\geq_n, \leq_n \mid n \in \mathbb{N}\})$
- The DL admits only the connective $\sqsubseteq$ applicable to both concepts and roles, i.e., $C_{DL}=\{\sqsubseteq, \sqsubseteq_R\}$

Then this DL is not AGM-compliant.

Proof

Set $K=\{R \sqsubseteq S\}$, $X=\{x \in L \mid Cn(\{x\}) \subset Cn(K)\}$, as in lemma 5.4.

We define two interpretations $I=(\Delta^I,\cdot^I)$, $I'=(\Delta^{I'},\cdot^{I'})$, as follows:

$\Delta^I=\Delta^{I'}=\{a_1,a_2,b_1,b_2,c\}$

$B^I=B^{I'}=\varnothing$ for all concepts B

$y^I=y^{I'}=c$ for all individuals y

$R_0^I=R_0^{I'}=\varnothing$ for all roles R_0, other than R, S

$R^I=R^{I'}=\{(a_1,b_1), (b_1,a_1), (a_2,b_2), (b_2,a_2)\}$

$S^I=\{(a_1,b_1), (b_1,a_1), (a_2,b_2), (b_2,a_2)\}$

$S^{I'}=\{(a_1,b_2), (b_2,a_1), (a_2,b_1), (b_1,a_2)\}$

Notice that the two interpretations differ only in the interpretation of the role S.

These interpretations are so close that for any (arbitrarily complex) concept term C formed using the operators in the hypothesis, it holds that $C^I=C^{I'}$.

This can be shown by applying induction on the number of operators that appear in C. If the role S does not appear in C, then the result is obvious; if it does, the definition of S^I and $S^{I'}$ make it easy to show the result; a detailed proof of this fact is rather lengthy and technical and is omitted.

A direct consequence of this result is that any axiom involving concept terms is satisfied by I iff it is satisfied by I′.

Furthermore, using induction again, we can also show that all axioms in X that involve role terms are actually tautological, i.e., they belong to $Cn(\varnothing)$. The proof to this fact is based on the fact that the only operator involving roles that is allowed in the DL of the hypothesis is the inverse operator $^{-}$. Any axiom of the form "$R_1 \sqsubseteq R_2$" for R_1, R_2 role terms that is implied by $\{R \sqsubseteq S\}$ will be either tautological, or equivalent to $\{R \sqsubseteq S\}$. Thus, all axioms involving roles in X are tautological. A detailed proof of this fact uses induction and is omitted.

Now, it is easy to notice that $(\Delta^I,\cdot^I)$ satisfies K (because $R^I \subseteq S^I$).

By the definition of X, we get that $K \vdash X$, so X is also satisfied by $(\Delta^I,\cdot^I)$.

Now take any axiom in X: if this axiom involves role terms then it is tautological so it is satisfied by $(\Delta^{I'},\cdot^{I'})$; if it involves concept terms then, by the result above it is satisfied by $(\Delta^{I'},\cdot^{I'})$, because it is satisfied by $(\Delta^{I},\cdot^{I})$. Therefore all axioms in X are satisfied by $(\Delta^{I'},\cdot^{I'})$, so X is satisfied by $(\Delta^{I'},\cdot^{I'})$.
On the other hand $R^{I'} \not\subseteq S^{I'}$, so K is not satisfied by $(\Delta^{I'},\cdot^{I'})$.
Thus, we found an interpretation, namely $(\Delta^{I'},\cdot^{I'})$, which satisfies X but not K, so $X \not\models K$.
Therefore: $Cn(X) \subset Cn(K)$.
To complete the proof, we need to show that $Cn(\varnothing) \subset Cn(X)$. Given that at least one of the operators $\forall$, $\exists$, $(\geq_n)$, $(\leq_n)$ (for some n) appears in the DL, we conclude that at least one of the facts: $\forall R.A \sqsupseteq \forall S.A$ or $\exists R.A \sqsubseteq S.A$ or $(\geq_n R.A) \sqsubseteq (\geq_n S.A)$ or $(\leq_n R.A) \sqsupseteq (\leq_n S.A)$ will appear in X, because they are implied by $R \sqsubseteq S$. None of these axioms is tautological (as can be easily shown), so X is not tautological either.
These facts, combined with lemma 5.4, conclude the proof. □

This negative result persists if the DL under question admits $\equiv$ (applicable to both concepts and roles) instead of $\sqsubseteq$, or if it admits both connectives. The same result can be shown if we add any connective from those in table 5.1 except non-proper-inclusion ($/\sqsubset$) for roles, transitive roles (the axiom Trans(.)), axioms with individuals, functional-only roles and/or qualified number restrictions. It also persists if we add the concept existence connective ($\exists_{\#}$), introduced in [HPS04], or the connectives that allow more general cardinality restrictions on concepts ($\leq^C_n$, $\geq^C_n$), introduced in [BBH96]. These results are summarized in the following theorem:

Theorem 5.4. Consider a DL <L,Cn> with the following properties:

- The DL admits at least two role names (say R, S) and one concept name (say A), i.e., $N_{DL}{}^R \supseteq \{R, S\}$, $N_{DL}{}^C \supseteq \{A\}$
- The DL admits at least one of the operators $\forall$, $\exists$ (full or limited), $(\geq_n)$, $(\leq_n)$, $(\geq^Q_n)$, $(\leq^Q_n)$ for at least some n, i.e., $O_{DL} \cap (\{\forall, \exists\} \cup \{\geq_n, \geq^Q_n, \leq_n, \leq^Q_n \mid n \in \mathbb{N}\}) \neq \varnothing$
- The DL admits any (or none) of the operators $\neg$ (full or atomic), $\sqcap$, $\sqcup$, $^-$, $\top$, $\bot$, $\{\ldots\}$, functional roles ^{f}, in addition to any of the operators of the previous bullet, i.e., $O_{DL} \subseteq \{\neg, \sqcap, \sqcup, ^-, \top, \bot, \{\ldots\}, ^f, \forall, \exists\} \cup \{\geq_n, \geq^Q_n, \leq_n, \leq^Q_n \mid n \in \mathbb{N}\}$
- The DL admits the connectives $\sqsubseteq$ and/or $\equiv$ applicable to both concepts and roles, any additional connective from table 5.1 (except $/\sqsubset$ for roles) and (possibly) the transitivity axiom Trans(.), the concept existence connective ($\exists_{\#}$) or cardinality restrictions on concepts ($\leq^C_n$, $\geq^C_n$), i.e., $C_{DL} \supseteq \{\sqsubseteq, \sqsubseteq_R\}$ or $C_{DL} \supseteq \{\equiv, \equiv_R\}$ and C_{DL} contains any of the axioms in table 5.1 (except $/\sqsubset$ for roles), plus, possibly, Trans(.), $\exists_{\#}$, $\leq^C_n$, $\geq^C_n$.

Then this DL is not AGM-compliant.

Proof

The proof is along the lines of the proof of theorem 5.3, using the same argumentation.
More specifically, if the axiom $R \sqsubseteq S$ is allowed in the DL at hand take $K=\{R \sqsubseteq S\}$; otherwise take $K=\{R \equiv S\}$, where R, S are roles in the namespace of the DLs. Define X as in theorem 5.3 and take the same interpretations; the same result can be shown regarding concept terms, despite the additional operators that are allowed in this theorem. Furthermore, as in theorem 5.3, all axioms that involve roles and belong to X are tautological, despite the addition of the new axiom types for roles. The rest of the proof is identical to the proof of theorem 5.3 and is omitted. □

Discussion

As was already mentioned in section 5.3, the determination of AGM-compliance for a DL should be made on a per-case basis, as there is no single operator and/or connective that guarantees AGM-compliance. As was shown in this section, the same is true for the determination of non-AGM-compliance.

Despite the validity of this general remark, the results of the last two sections uncovered a certain pattern that leads to a very important empirical rule regarding AGM-compliance in DLs under OVA: if we can define the transformations required by theorem 5.2, then the DL is AGM-compliant; if not, then there is good chance that lemma 5.4 can be applied in a set of the form $\{R \sqsubseteq S\}$ or $\{R \equiv S\}$, for two roles R, S in the DL namespace. This provides a simple test to determine whether a DL is AGM-compliant or not.

Our analysis also shows that role operators, especially role intersection $\sqcap_R$ and role complement $\neg_R$, play an important role in the AGM-compliance of DLs. If these operators exist, then, in most cases, the transformations required by theorem 5.2 are definable, so the DL is AGM-compliant. If they don't exist, lemma 5.4 is usually applicable, so the DL is not AGM-compliant. If no expressive role terms can be defined, then (in most cases) the DL at hand cannot admit AGM-compliant contraction operations upon role axioms; in effect, role axioms become very coarse-grained, making it impossible for a contraction operator to handle a contraction upon sets containing such axioms in an AGM-compliant way.

For this reason, we highly encourage research on DLs that admit these operators due to their nice behavior with respect to contraction. Unfortunately, very few logics with role operators have been studied in the literature (notable exceptions being [HS00], [LS00]), so the computational overhead caused by such operators is largely unknown. The initial results indicate that most such logics have a computational worst-case complexity of NEXPTIME [LS00], [Zol06].

It should be stressed that the empirical rule presented in this subsection is not always applicable, as it may break down in certain artificial examples. However, its practical applicability is unquestionable, as it turns out to hold in most interesting DLs. This will be verified by our analysis in section 5.6, as well as by the categorization of the DLs that have been studied in the literature to AGM-compliant and non-AGM-compliant ones (see table 5.2 in section 5.6).

5.5. OWL and AGM-Compliance

Introduction

As already mentioned, OWL [DSBHH+04] is one of the most important languages used for ontological representation in modern ontologies; its importance is emphasized by the fact that it has become a W3C Recommendation. OWL uses an RDF-like syntax [MSB06], but its semantics is very close to the DL semantics. It comes in three different "flavors" (namely OWL Full, OWL DL and OWL Lite), each with different expressive power and reasoning complexity. Apart from its purely "logical" features, OWL also allows certain meta-logical constructs, stating, for example, the author of the ontology, its version, external ontologies upon which the ontology is depending, compatibility information, comments and other human-readable information. Such features are collectively referred to as annotation features.

Annotation features are generally viewed as secondary properties of the language [HPS04]. They are very special in the sense that they contain human-readable information, as opposed to machine-readable information which is contained in the "logical" part of the ontology. Also, the annotation features cannot be described using DL axioms, while the "logical" part of OWL DL and OWL Lite can [HPS04]. OWL Full (with or without annotations) is also special in this respect, because it provides a more complete integration with RDF, using features not available in DLs which render it undecidable [HPS04].

The fact that annotation features force a departure from the standard DL model (being non-expressible using DL axioms) is not a problem for our analysis, because all flavors of OWL, with or without annotation features, can be easily engulfed by the general logical framework of Tarski that we use in this work. Therefore, we can use our general theory of AGM-compliance to study these logics. Due to the differences in semantics caused by the introduction (or not) of the annotation features in OWL, we will study these two different cases independently.

OWL DL and OWL Lite Without Annotation Features

As already mentioned, OWL DL and OWL Lite (without annotation features) have been shown equivalent (in terms of expressiveness) with the DLs $SHOIN^+(D)$ and $SHIF^+(D)$ respectively [HPS04]. These are very expressive DLs, allowing several different operators and connectives, as well as *datatypes*.

More specifically, $SHOIN^+(D)$ allows for the following operators: $\top$, $\bot$, $\sqcap$, $\sqcup$, $\neg$, $\{\ldots\}$, $^-$, $\exists$, $\forall$, $(\geq_n)$, $(\leq_n)$. In addition, it allows a datatype theory (D), which is a mapping from a set of datatypes to a set of values plus a mapping from data values to their denotation (see [HPS04] for details). To make datatypes useful, the logic also allows datatype (or concrete) roles, which are binary relationships between individuals and data values, as well as the operators $\exists$, $\forall$, $(\geq_n)$, $(\leq_n)$ for datatype roles and the operator $\{\ldots\}$ for data values. The term abstract roles is used to denote the standard DL roles which are binary relationships between individuals.

The axioms allowed in this logic are concept hierarchies, abstract role hierarchies and datatype role hierarchies; individual inclusion, equality and inequality; role transitivity (for abstract roles only); and the concept existence axiom [HPS04]. The concept existence axiom (denoted by $\exists_\#$) asserts that a concept term is interpreted to be a non-empty concept.

$SHIF^+(D)$ is very similar but less expressive than $SHOIN^+(D)$; $SHIF^+(D)$ is just $SHOIN^+(D)$ without the $\{\ldots\}$ constructor and with the at-least and at-most constructors limited to 0 and 1.

It can be noticed that both $SHOIN^+(D)$ and $SHIF^+(D)$ allow for abstract role hierarchies without supporting role operators ($\neg_R$, $\sqcap_R$). This motivates us to use the general empirical rule we introduced before and apply lemma 5.4 to the set $\{R \sqsubseteq S\}$ for two abstract roles R, S. It turns out that this approach allows us to show $SHOIN^+(D)$ and $SHIF^+(D)$ to be non-AGM-compliant:

Lemma 5.5. The DLs $SHOIN^+(D)$ and $SHIF^+(D)$ are not AGM-compliant, provided that they admit at least two roles in their namespace.

Proof

We will use a variation of the proof of theorem 5.4.

Let's start with $SHOIN^+(D)$. We notice that $SHOIN^+(D)$ allows datatype roles, operators and connectives which are not supported by theorem 5.4. However, this problem can be easily taken care of by replacing the two interpretations of the proof of the theorem $(\Delta^I, \cdot^I)$ and

$(\Delta^{I'},\cdot^{I'})$, by the interpretations $(\Delta^{I},\cdot^{I},\Delta_D{}^{I})$ and $(\Delta^{I'},\cdot^{I'},\Delta_D{}^{I'})$ which have the same domain and interpretation function as the original I, I′ but additionally contain a datatype domain. If we define the datatype domains of the two interpretations I, I′ to be identical and interpret all data values and datatype roles in an identical way in both interpretations, then the steps of the proof can be repeated without problems.
Similar steps will give us the proof that $SHIF^+(D)$ is not AGM-compliant. □

Corollary 5.4. OWL DL and OWL Lite without their annotation features are not AGM-compliant.
Proof
Using the mapping defined in [HPS04] between OWL DL and $SHOIN^+(D)$, it becomes obvious that the two logics are AGM-equivalent, so by lemma 5.5, OWL DL is non-AGM-compliant. Similarly, OWL Lite is not AGM-compliant. □

Corollary 5.4 shows that the interesting flavors of OWL are non-AGM-compliant (OWL Full is considered non-interesting because its inference problem is undecidable). The latter result assumes that OWL allows at least two role names in its namespace, which is hardly an assumption given OWL's definition [DSBHH+04].

OWL With Annotation Features

But what if annotation features are included in OWL? Does this make the situation any better? Unfortunately not: the annotation features are meant to be read by humans, so they carry no special meaning (they imply nothing). There is one exception though: the owl:imports annotation feature carries some meaning for the parser, making it substantially different from the other annotation constructs. More specifically, owl:imports is a meta-logical annotation property forcing the parser to include another ontology in the current ontology. The fact that this axiom is machine-readable has substantial consequences in reasoning and AGM-compliance. So, let us first consider the other, simpler annotation features.

The fact that annotation features carry no logical, machine-readable information can be formally expressed by saying that an annotation axiom $x \in L$ of a DL $<L,Cn>$ implies nothing but itself; this can be expressed by the condition: $Cn(K \cup \{x\}) = Cn(K) \cup \{x\}$ for all $K \subseteq L$ and all annotation axioms $x \in L$. Furthermore, an annotation axiom $x \in L$ cannot be implied by any other set of propositions, unless this set of propositions contains x itself, i.e., for all $K \subseteq L$ and all annotation axioms $x \in L$, $x \in Cn(K)$ iff $x \in K$. These two conditions can be extended to sets $X \subseteq L$ of annotation axioms.

So, suppose that we start with a logic (or DL) $<L_1,Cn_1>$, and add certain axiom types to get a richer logic (DL) $<L_2,Cn_2>$; if these axiom types are annotation features, then, by the above analysis, the following properties should hold:

- $L_1 \subseteq L_2$
- For all $X \subseteq L_1$ it holds that $Cn_1(X) = Cn_2(X)$
- For all $X \subseteq L_1$ and all $Y \subseteq L_2 \backslash L_1$ it holds that $Cn_2(X \cup Y) = Cn_2(X) \cup Y$
- For all $Z \subseteq L_2$ and all $Y \subseteq L_2 \backslash L_1$ it holds that $Y \subseteq Cn_2(Z)$ iff $Y \subseteq Z$

These properties make corollary 4.11 applicable in the situation, which verifies that the resulting logic $<L_2,Cn_2>$ is AGM-compliant iff the original logic $<L_1,Cn_1>$ is. So, starting with OWL (DL or Lite) and adding annotation features does not change the non-AGM-compliant status of OWL (DL or Lite). Thus:

Corollary 5.5. OWL DL and OWL Lite with annotation features (but not owl:imports) are not AGM-compliant.

OWL With "owl:imports(.)"

For the annotation feature owl:imports things are very different. The axiom owl:imports(O) has several consequences except from itself: it implies anything that is implied by the ontology (DL KB) O. Thus, corollary 4.11, and, consequently, corollary 5.5, are no longer applicable because their prerequisites fail for this annotation feature; a different method is required to address this problem.

Let us study this axiom a little closer. Consider the set K={owl:imports(O)}; the DL KB K forces the parser to include O in K dynamically, as a pre-processing step before any kind of reasoning upon K. This process involves replacing owl:imports(O) with all the axioms of O. This may cause one to believe that owl:imports(.) adds nothing to the expressive power of the logic: whenever an owl:imports(O) axiom is encountered, it could simply be replaced by O, eliminating the need for a special axiom.

Unfortunately, there is a problem with this approach. If owl:imports(O) is replaced by O in such a manner, the replacement is static; it is made at design-time, instead of run-time. If O is subsequently modified, then the reasoning made upon K will still reflect the old status of O, instead of the new one. For reasons explained in previous chapters, we cannot reasonably assume that any ontology in the Semantic Web will remain static and unchanged through the course of time, so the problem is very real. As a consequence, owl:imports(.) cannot be replaced by the axioms of O. Notice that this would not be an issue if we could somehow guarantee that O would remain static and unchanged; however, given the dynamic nature of the Web, such an assumption would be highly unrealistic.

But then, what are the consequences of K? Obviously K implies itself, as well as all the consequences of O and nothing else; thus: $Cn(K)=Cn(O)\cup K$. On the other hand, owl:imports cannot be replaced by the axioms of O, for reasons explained before. Therefore: $K\not\subseteq Cn(O)$, unless $K\subseteq O$. In other words, the axiom owl:imports(O) cannot be obtained as a consequence of any ontology, unless it is contained explicitly in the ontology.

Rephrasing this fact in the formal terminology we use in this dissertation we get that $Cn(K)\supset Cn(O)$. This fact allows us to show the following proposition:

Proposition 5.1. Consider any logic <L,Cn> that allows axioms of the form owl:imports(.) with the semantics described above. If there is at least one $x_0\in L$ such that $Cn(\{x_0\})\neq Cn(\emptyset)$ and x_0 is not of the form owl:imports(.), then <L,Cn> is non-AGM-compliant.

Proof

Set $K'=\{x_0\}$, $K=\{$owl:imports$(K')\}$.

Then, by the hypotheses of the proposition and the previous analysis, the following holds:

$Cn(\emptyset)\subset Cn(K')\subset Cn(K)=Cn(K')\cup K$.

Take any $X\subseteq L$, such that $Cn(X)\subset Cn(K)=Cn(K')\cup K$.

If $X\cap K\neq\emptyset$ then $X\supseteq K$ so $Cn(X)\supseteq Cn(K)$ a contradiction.

So $X\cap K=\emptyset$ and, since $X\subseteq Cn(X)\subset Cn(K')\cup K$ it follows that $X\subseteq Cn(K') \Rightarrow Cn(X)\subseteq Cn(K')$.

These two results, combined with definition 4.5, imply that the family $\wp=\{K'\}$ is a cut of K; moreover $Cn(\cap_{Y\in\wp}Cn(Y))=Cn(K')\neq Cn(\emptyset)$.

We conclude that <L,Cn> is not AGM-compliant. □

Proposition 5.1 is rather catastrophic for the purposes of AGM-compliance in logics which support dynamic linking of KBs (in the sense of owl:imports). In effect, all non-trivial logics that support dynamic linking are non-AGM-compliant, unless we could somehow guarantee that the objects upon which dynamic linking is made are static. This assumption is highly unrealistic in practically any context, so we could safely assume that it does not hold. Therefore, all flavors of OWL as well as any other non-trivial fragment of OWL that allows for axioms of the form owl:imports is non-AGM-compliant:

Corollary 5.6. Any flavor of OWL with or without the general annotation features that allows the owl:imports axiom is non-AGM-compliant.

5.6. Discussion

Application to DLs in the Literature

In the previous sections we studied the AGM-compliance of several DLs at a fairly abstract level; we did not focus on any specific DL, nor enforced any special properties on the DLs considered, except, of course, from the properties needed to acquire our results. In effect, we dealt with the DL family as a whole, including DLs that have not yet been considered in the literature. This approach allows our results to be of use to researchers who develop new DLs; if the focus is on developing a DL that can be rationally updated, then AGM-compliance should be a desirable feature of the new DL, along with high expressive power, low reasoning complexity etc.

However, the results of this chapter can be easily applied to several DLs that have already been considered in the literature. In this subsection, we will provide an indicative (but not necessarily complete) list of DLs (see table 5.2) for which a definite answer regarding AGM-compliance can be given using the results appearing in this dissertation. This list is easily obtainable by the results of this chapter; no formal proofs will be given as these are straightforward. For a definition of the logics below, refer to [BS01], [BCMNPS02], [BHS03], [DSBHH$^+$04], [HPS04], [HS01], [HS00], [LS00]. This list refers to logics under the OVA model, as for CVA things are very straightforward (see corollary 5.1).

All DLs between SH and SHOIQ can be shown to be non-AGM-compliant; this family contains several DLs, like SHI, SHIF, SHOIF, SHOIN, SHOIQ, etc. All these logics admit role hierarchies but not role operators, so these results are actually special cases of theorem 5.4. For similar reasons, adding role hierarchies to the AL family leads to non-AGM-compliance; that is, FL_0 and FL^- with axioms involving role terms and all DLs between ALH and ALHCIOQ are non-AGM-compliant. This family includes several logics, such as ALHE, ALHNC, etc.

The addition of the concept existence axiom ([HPS04]) to these logics does not change this fact: SHOIN$^+$ and SHIF$^+$, for example, are not AGM-compliant (theorem 5.4). Similarly, non-AGM-compliance persists upon the addition of datatypes (see the proof of lemma 5.5), so SHOIN(D), SHOIN$^+$(D), SHIF(D), SHIF$^+$(D) are not AGM-compliant either.

Finally, as was shown in corollary 5.4, corollary 5.5 and corollary 5.6 none of the three flavors of OWL is AGM-compliant if the owl:imports axiom is included; OWL DL and OWL Lite without owl:imports and with or without the other annotation features are non-AGM-compliant either.

Description Logic(s)	AGM-compliant?	Relevant Result
$ALCO^{\neg,\sqcap,\sqcup}$	YES	Corollary 5.3
$ALCO^{(\neg),\sqcup}$ with no axioms involving role terms	YES	Theorem 5.2
$ALC^{\neg,\sqcap,\sqcup}$ with empty Abox	YES	Theorem 5.2
$ALC^{(\neg),\sqcup}$ with empty Abox and no axioms involving role terms	YES	Theorem 5.2
ALCO with an infinite number of role names and no axioms involving role terms	YES	Theorem 5.2
ALC with an infinite number of role names, empty ABox and no axioms involving role terms	YES	Theorem 5.2
All DLs with more operators (but no more axiom types) than the above	YES	Theorem 5.2
All DLs between SH and SHOIQ	NO	Theorem 5.4
All DLs between ALH and ALHCIOQ	NO	Theorem 5.4
FL_0, FL^- with axioms involving role terms	NO	Theorem 5.4
$SHOIN^+$, $SHIF^+$	NO	Theorem 5.4
SHOIN(D), $SHOIN^+$(D), SHIF(D), $SHIF^+$(D)	NO	Lemma 5.5
OWL DL, OWL Lite without annotation features	NO	Corollary 5.4
OWL DL, OWL Lite with annotation features but not owl:imports	NO	Corollary 5.5
OWL Full, OWL DL, OWL Lite with the owl:imports axiom and with or without annotation features	NO	Corollary 5.6

Table 5.2. AGM-compliant and non-AGM-compliant DLs

The addition of role operators to the AL family results in some AGM-compliant DLs. In particular, $ALCO^{\neg,\sqcap,\sqcup}$ is AGM-compliant (see corollary 5.3); if no axioms involving role terms are allowed, then $ALCO^{(\neg),\sqcup}$ and $ALCO^{(\neg),\sqcap,\sqcup}$ are also AGM-compliant, because the transformations need not support role axioms, so they are definable (see theorem 5.2). For similar reasons, $ALC^{\neg,\sqcap,\sqcup}$ with empty Abox, as well as $ALC^{(\neg),\sqcup}$ and $ALC^{(\neg),\sqcap,\sqcup}$ with empty Abox and no axioms involving role terms are AGM-compliant. For reasons explained in section 5.3, AGM-compliance persists if we add more operators (but not new axiom types) to any of these logics; for example, all DLs with more operators than $ALC^{\neg,\sqcap,\sqcup}$ and no Abox (such as ALB) are AGM-compliant.

If we have an infinite pool of role names, we can use the alternative transformation (T_2') that does not use the constant $\top_R$. This way, ALC (and all languages with more operators than ALC) become AGM-compliant, provided that no axioms involving role terms are included and that the Abox is empty. Similarly, all DLs with more operators than ALCO are AGM-compliant if they do not allow axioms involving roles.

The latter results show that there are AGM-compliant DLs which don't satisfy the AGM assumptions (like ALC, ALCO etc). Moreover, they further indicate the aforementioned

importance of role operators (role intersection and complement) for AGM-compliance: in most cases, it is the absence of these operators that bars AGM-compliance. All the results of this subsection can be found in a more compact form in table 5.2.

DLs and Base-AGM-Compliance

The results of this chapter showed that the situation is rather diverse regarding the AGM-compliance of DLs: certain logics under the OVA model are AGM-compliant, while others, including all non-trivial DLs under the CVA model, are non-AGM-compliant. For base-AGM-compliance, things are not so diverse; in fact, they are very discouraging, as expected by proposition 4.9. This is formally expressed in the following proposition:

Proposition 5.2. Any DL (under CVA or OVA) that is at least as expressive as FL_0 and whose namespace allows at least two concept names and one role name is non-base-AGM-compliant.

Proof

Take any DL <L,Cn> with the properties of the proposition.

If the DL follows the CVA model, the result is a direct consequence of corollary 5.1 and corollary 4.13.

If the DL follows the OVA model, then (without loss of generality) consider two concept names A, $B \in N_{DL}^{C}$ and one role name $R \in N_{DL}^{R}$.

FL_0 allows concept equivalence ($\equiv$) and value restriction ($\forall$), among other things, thus:

$\{A \equiv B\}, \{\forall R.A \equiv \forall R.B\} \subseteq L$

Using the model-theoretic semantics of DLs, it is trivial to show that:

$Cn(\varnothing) \subset Cn(\{\forall R.A \equiv \forall R.B\}) \subset Cn(\{A \equiv B\})$, so, by proposition 4.9, <L,Cn> is non-base-AGM-compliant. □

As already mentioned in section 5.2, the DL FL_0 represents the least expressive DL worth studying; thus, the latter result implies that all interesting DLs are actually non-base-AGM-compliant.

Topology of AGM-Compliance with DLs and OWL

The results appearing in this chapter can be summarized in figure 5.2 (see next page), which improves figure 4.18 by the addition of DLs and OWL. As we can see in the figure, all DLs under CVA are non-AGM-compliant (theorem 5.1); the class of DLs under the OVA model intersects the AGM class [Bor96] and the class of AGM-compliant logics (theorem 5.2), but there are also DLs under OVA which are non-AGM-compliant (theorem 5.4). Each of the OWL flavors can be viewed under either model (OVA or CVA) and with or without annotations and/or the "owl:imports" axiom. The addition of annotations, as well as some features of OWL Full, cause a departure from the standard DL model, so the variations of OWL do not form a subset of the DL family; all such variations are non-AGM-compliant (corollary 5.4, corollary 5.5, corollary 5.6). Finally, all these logics (DLs and OWL) are non-base-AGM-compliant (proposition 5.2).

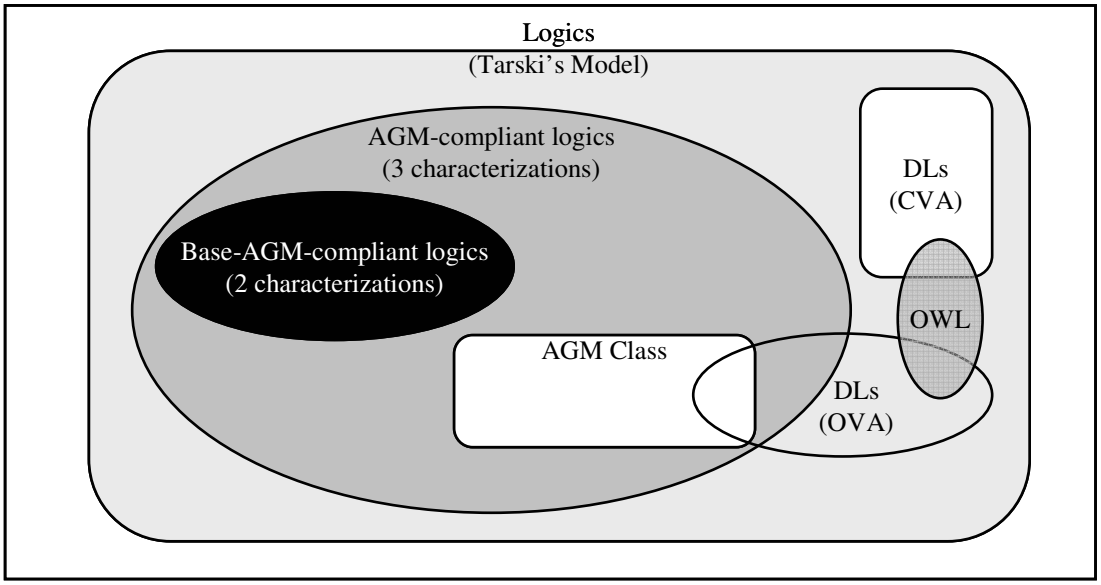

Figure 5.2. Topology of AGM-compliance with DLs and OWL

Chapter 6: Conclusion and Future Work

In theory, there is no difference between theory and practice; in practice, there is.

Chuck Reid

6.1. Belief Change and Ontology Change

Research Path

In this dissertation, we studied the effects of applying the theory of belief change [Gar92a] to the general problem of ontology change with emphasis on ontology evolution [HS04]. Initially, we proposed a certain terminology regarding ontology change and performed a study on the interaction and overlap between the different fields that have emerged in relation to this problem (section 2.2).

Then, it was shown that the general ideas developed in the belief change literature can be used (at least) as an inspiration for the development of similar ideas in the field of ontology evolution. Important belief change concepts, like the discrimination between foundational and coherence theories [Gar92b], the Principle of Minimal Change [KM90b], the different semantics that a change operation can take [FH94], [KM90a] and others can be used in our quest for a deep understanding of the formal processes behind ontology evolution (section 3.3). Unfortunately, such issues have been neglected in the, yet immature [NK04], ontology evolution field. We believe that the rich literature on belief change will help in providing the necessary maturity to the field and protect ontology evolution researchers from potential pitfalls, as well as point them to the most appropriate methods to solve problems whose counterparts have already been addressed in belief change.

Our approach provided the necessary formalizations allowing an ontology evolution approach that will hopefully resolve the major weaknesses of the field, which were identified in section 3.2. These are the lack of automatic methods for ontology evolution, the great number of different operations that are allowable upon an ontology and the lack of adequate formalizations of the problem. Our study in sections 3.3 and 3.4 showed that the use of belief change techniques in the ontology evolution field is feasible: most of the concepts used in belief change can be easily migrated to the ontology evolution context, because of the close connection between the two research areas. Whenever complications arise, these are mainly due to the different knowledge representation formalisms that are being used in the different fields rather than the underlying intuitions.

This part of our work was not intended to provide concrete solutions to the problem of ontology evolution; rather, we aimed at providing some general guidelines towards a research

direction that will ultimately lead to the desired solutions, as well as to set stable formal foundations upon which such solutions could be based.

AGM-Compliance in Description Logics and OWL

In our attempt to evaluate our approach we attempted the migration of the AGM theory [AGM85] to the world of ontologies. This migration exploited the solid foundations set by the study we performed in chapter 3 and showed how these abstract intuitions can be used to form more concrete approaches to the problem of ontology evolution.

As was argued, the AGM theory is the most influential approach in the field of belief change, because it properly captures common intuition regarding the concept of rationality and has several interesting theoretical properties (section 2.5). Thus, it formed a reasonable initial choice for applying the general research method proposed in chapter 3 (see section 4.2 for motivation and details).

Our migration attempt was initiated by relaxing the AGM assumptions in order to show that the AGM theory can be reformulated in a vastly more general context, being applicable to all logics under the logical framework of Tarski (section 4.3). Certain problems were associated with this generalization, which were studied and evaluated in sections 4.4 and 4.6.

The most important problem we faced was the inability of certain logics under Tarski's framework to accommodate an operation satisfying the AGM postulates. We provided three different equivalent characterizations of logics that admit such an operation, termed AGM-compliant logics. These characterizations were based on the notions of decomposability (definition 4.2, definition 4.3 and theorem 4.1, appearing in section 4.4), cuts (definition 4.5 and theorem 4.2, appearing in section 4.6) and max-cuts (definition 4.6 and theorem 4.3, appearing in section 4.6).

We showed that this problem is caused by the introduction of the recovery postulate in the contraction postulates (lemma 4.1), which is a highly debated postulate anyway [Mak87], [Han99b]. We provided several interesting theoretical results related to the AGM theory of contraction, like its relation with foundational belief change (section 4.8), as well as the connection of AGM-compliance with the concept of negation (definition 4.7, in section 4.7), the rule of introduction of disjunctions in the premises, or IDP rule, (corollary 4.8, in section 4.7) and the concept of roots (section 4.9). Finally, the relation of Tarski's model with lattice theory was studied in section 4.5 under the light of our results on the generalization of the AGM theory; this study showed that the AGM-compliance of a logic is a property that can be completely determined by the structure of the (unique, modulo equivalence) lattice that represents it (corollary 4.3 and corollary 4.4).

Subsequently, this abstract study was specialized so as to focus on the logics that are most commonly used for ontological representation, namely DLs and OWL. Our analysis provided specialized conditions for AGM-compliance (section 5.3) and non-AGM-compliance (sections 5.2 and 5.4) which are applicable for such logics.

A corollary of our study was that OWL does not support the AGM postulates, so the approach is not useful for OWL ontologies (corollary 5.4, corollary 5.5, corollary 5.6 and proposition 5.1, appearing in section 5.5). Regarding DLs, the situation is much better: there are certain DLs in which an AGM-compliant contraction operator can be defined (theorem 5.2 and its corollaries in section 5.3), as well as several non-AGM-compliant DLs (theorem 5.1, theorem 5.3 and theorem 5.4, appearing in sections 5.2 and 5.4). Our results do not currently provide a complete characterization of AGM-compliance for DLs; this is an important goal

for our future work. A side-effect of our work was the discrimination between two closely related knowledge representation models for DLs, namely CVA and OVA (section 3.3) and their effects on AGM-compliance (sections 5.2, 5.3 and 5.4), as well as the introduction of an AGM-compliant contraction operator for AGM-compliant DLs.

These results are summarized in figure 5.2 (page 156) and in table 5.2 (page 154). Unfortunately, many interesting DLs were shown to be non-AGM-compliant, mainly due to the recovery postulate (lemma 4.1). This negative result and the objections that have appeared in the literature against this postulate [Fuhr91], [Han96], [Han99b] may motivate future work on a "replacement" of this postulate, or on some approximation of it, that would properly capture the Principle of Minimal Change in addition to being applicable to non-AGM-compliant logics (and DLs).

As far as AGM-compliant DLs are concerned, we believe that research on ontology evolution should use the feature of AGM-compliance, thus taking advantage of the numerous results that have appeared in the literature on belief change and the AGM theory during the past twenty years. We believe that AGM-compliance constitutes a desirable property that any rationally updatable DL (or logic, in general) should possess.

For this reason, we plan to continue our research on the application of the AGM theory to the DLs that support it. In this respect, notice that AGM-compliance simply guarantees the existence of a contraction operator that satisfies the basic AGM postulates for contraction; one of our future goals is to determine the relation of AGM-compliance to other results related to the AGM theory, such as the various representation theorems (e.g., [AGM85], [AM85], [Gar92a], [GM88], [Gro88], [KM90b]), the supplementary postulates of contraction [AGM85], the Levi and Harper identities [Gar92a] and the AGM theory of revision [AGM85].

6.2. Main Contributions of our Research

Contributions to the Field of Belief Change

This thesis provided a theoretical framework that bridged two different research fields, namely belief change and ontology evolution. As a result, research on both fields may benefit from this work.

The most important contribution of our work in the field of belief change is the notion of decomposability. As mentioned in section 4.4, decomposability forms another equivalent representation result for the AGM postulates of contraction. Several similar results have already appeared in the literature ([AGM85], [AM85], [GM88], [Gro88], [KM90b]); however, their applicability is restricted to classical logics in all cases. Our characterization is unique in the sense that it can be applied to a far larger class.

Another contribution is related to the fact that our approach views the field of belief change in general, and the AGM theory in particular, under a more general prism that covers many more logics than the ones usually considered in belief change. This is important, because the applicability of most belief change theories has been, up to now, minimal, compared to the importance of the problem and the advances made in the field (see, e.g., [Kel98], [Wil98] for some such applications). We believe that this is mainly due to the relatively narrow logical framework in which belief change theories are being developed.

Our approach showed the path towards the generalization and adaptation of results from this field to other logics that are used in interesting modern applications (in our case, to DLs and OWL used in ontologies and the Semantic Web). We are aware of very few similar attempts to apply belief change theories to non-classical logics or contexts ([BAGM99], [Foo95], [KL04], [LM04], [MLB05], [Wass98], [WF99]). We believe that the problems encountered during our migration attempt are typical of the problems that would be faced by other researchers attempting similar migrations of belief change theories to non-classical logics and contexts. Thus, the experience gained by this work will prove helpful in future migration attempts.

Contributions to the Fields of Ontology Change and Ontology Evolution

Our attempt for a clarification of the various understandings of different terms related to ontology change is the first contribution of our work as far as ontology change is concerned. This study set the boundaries of each of the related research fields and studied their relation, interaction and overlap. This work is important because most of these terms are being used in different ways by different researchers, causing a certain amount of confusion to a newcomer.

Moreover, our work employed an original approach to the problem of ontology evolution, opening an innovative research path that may resolve several of the problems currently faced by ontology evolution researchers. This approach is based on belief change, thus it views the problem under a totally different standpoint and uses different paradigms. These two properties alone constitute a significant advancement in the field, because they encourage the development of alternative methods of addressing the problem.

Apart from this fact however, our analysis showed that approaches based on belief change perform better than current ontology evolution algorithms in several aspects, while, at the same time, allowing the use of current advances in the field of ontology evolution; this motivates us to use the best of both worlds in order to resolve the major weaknesses of the ontology evolution field (human participation, insufficient formalization, immaturity).

Furthermore, our fusion attempt revealed unknown parameters to the problem of ontology evolution, like the various principles and tradeoffs originally studied in the field of belief change. These may be used to improve the quality of currently employed ontology evolution algorithms.

Last but not least, our generalization of the AGM theory in chapter 4, provided the first postulation attempt for ontology evolution algorithms; this postulation determines the properties that any automatic ontology evolution algorithm should satisfy in order to behave in a satisfactory and adequate manner, in terms of rationality. The justification of this postulation's importance as well as its properties can be derived from the extensive research related to the AGM theory (which formed the basis for the postulation) that has been performed in the belief change literature.

Practical Contributions

Another very interesting question that needs to be addressed is related to the practical contributions of this work. The major innovation of this work in this respect is the introduction of the first fully automatic contraction algorithm for ontologies based on AGM-compliant DLs. This algorithm was based on syntactic considerations, which makes it very efficient in terms of computational complexity; however, a semantical analysis of the

properties of this algorithm is missing, an analysis which constitutes a significant goal for our future work.

Another interesting implication of our work is related to the explicit discrimination between two paradigms that are commonly employed in ontologies (CVA and OVA) as well as the study of their semantics and formal properties. This study showed that the assumption employed determines several of the formal properties of the logic at hand, including its inference mechanism and update semantics. To the best of the author's knowledge, this is the first explicit and formal reference to the differences between the two paradigms.

Moreover, our generalization of the AGM theory determined the applicability of this theory within a vast class containing all monotonic logics. The very nature of the AGM theory makes this generalization very important, because this theory determines the rationality of a change operation in a manner unrelated to the underlying logical formalism or context. Our results were not always positive, in the sense that there are several logics in which the AGM theory of contraction cannot be applied. On the other hand, our approach showed that there are also several logics, not originally considered by AGM, in which the AGM theory can be applied.

Furthermore, our analysis showed that it is the recovery postulate (K−6) which is responsible for the negative results. This fact provides a definite clue on the path that should be followed towards a weakened version of the AGM theory for non-AGM-compliant logics: by weakening the recovery postulate.

Other side-effects of this generalization include the experience gained in the migration and generalization process, which could be helpful in our future attempts to propose a weakening of the AGM theory or to generalize other theories introduced in the belief change literature so as to be applicable in the context of ontology evolution or elsewhere. Finally, this generalization showed that our proposal to generalize belief change theories to other contexts (such as ontology evolution) makes sense and is feasible, at least to some degree.

6.3. Concluding Remarks

Ontologies have found several applications in the Semantic Web, as well as in other areas of AI. Unfortunately, the majority of research studies in the area of ontology engineering have focused on construction issues [HS05a], with considerably less attention being given to the problem of changing an ontology in response to a certain need; in effect, ontologies have been so far mainly treated as being static [HS04]. However, there are several reasons why an ontology should change, the most important ones being changes in the domain and changes in the conceptualization of the domain. We have used the generic term ontology change to refer to any possible type of change required in an ontology in response to any possible need.

This dissertation can be divided in four parts: in the first part (chapter 2), we performed a shallow, but broad and critical literature review covering all the diverse types of ontology change. This allowed us to fix a certain terminology in an area that is plagued by the presence of underspecified terms which are used with different meanings by different researchers; as stated in the introduction, this is an important task for the field. In this part of our work we also reviewed the current state of the art in each field related to the broad area of ontology change and performed a short literature review on belief change, the AGM theory, DLs and OWL, which was useful for our purposes in subsequent chapters.

In the second part (chapter 3), we restricted our attention to the problem of ontology evolution and identified certain deficiencies of the methods that are currently used to tackle this problem. We argued that research on ontology evolution is still at a preliminary, immature stage and outlined the most important problems currently being faced in the field. We also explained why many of the practical applications of ontology evolution will require a different, more formal, computer-based approach to the problem.

To deal with these deficiencies, we proposed the use of old tricks to address this new problem. More specifically, we have presented a certain logic-based framework which allows the problem of ontology evolution to be formulated in terms of the more general problem of belief change, which has been extensively studied in the literature over the past twenty years. The proposed framework is not based on any additional assumptions except the standard assumptions made in the ontology evolution field, thus our reformulation is sound in the sense that it does not alter the properties of the problem in any way. It also allows the ideas and intuitions appearing in the rich belief change literature to be applied, with only minor modifications, to the ontology evolution problem. For this reason, we believe that our approach complements the currently used methods and could be used as a supplementary tool for ontology maintenance and evolution.

We argued that the solid theoretical foundations set by this study will allow the development of more concrete and sophisticated approaches to ontology evolution based on the notions studied by belief change researchers. Furthermore, we explained how the explicit description of the connections between the two fields and the theoretical foundations that were set in order to allow the migration of belief change ideas to ontology evolution will prove helpful in this direction.

The feasibility of this approach was evaluated in the third part of this dissertation (chapter 4), which can be viewed as an application, or a "case study", of our methodology. More specifically, we described a reformulation of the AGM theory in a very general context, applicable to all logics that can be engulfed by Tarski's logical model. This migration attempt did not focus on ontology evolution or ontology representation languages, but was formulated in a very general context. Despite that, it was exemplary of the research path we are proposing and showed the main problems that are likely to be encountered during the migration of belief change theories (in this case, the AGM theory) to other contexts (including, but not limited to, the context of ontology evolution). These problems are mostly related to the different properties of the representation languages used in the different areas rather than on the properties of the fields themselves; the properties of the dynamics of change are related to the knowledge level [New82], so they are not affected by the underlying representation language used. This study additionally led to several new results related to the AGM theory.

In the fourth part of this dissertation (chapter 5), the aforementioned study of the AGM theory was specialized in the context of DLs [BCMNPS02], one of the leading formalisms for ontological representation [BHS03], as well as in the context of OWL [DSBHH+04], a W3C Recommendation. This work led to specialized conditions related to the applicability of the AGM theory of contraction to these ontology representation languages, in effect determining the applicability of the AGM theory in the context of ontology evolution. Certain heuristics were also formulated allowing the determination of this applicability to logics not covered by this work.

In this dissertation, we started from a very general problem (i.e., applying belief change to ontology evolution) and gradually narrowed our focus to the applicability of a specific belief change theory (i.e., the AGM theory) to a general context (i.e., the logics under Tarski's model), followed by the application of the same theory (AGM theory) to the context of ontology evolution and ontology representation languages (i.e., DLs and OWL). We expect these results to allow the development of rational, formal, automatic, effective and efficient methods for ontology evolution. The gradual bounding of our focus led to results that may be applicable to diverse communities as well; for example, the generalized AGM theory could be useful in the study of belief change over languages not in the original focus of AGM, even if these languages are unrelated to the Semantic Web.

6.4. Future Work

It should be clear by the above analysis that this dissertation only scratched the surface of the relation between ontology evolution and belief change; much more work needs to be done on this issue, both on theoretical and on practical grounds. Our approach indicated that important theories from the belief change literature can be migrated, at least partially, to the world of ontologies; this generic proposal allows us to pursue several different paths of implementation, each using a different belief change approach for migration in the ontology evolution context. Not only the intuitions of the belief change research can be used in our quest for ontology evolution algorithms, but certain theories themselves could also prove helpful.

Our work was restricted to only one possible alternative of this more general research proposal, namely the migration of the AGM theory to the context of ontology evolution; the application of other belief change algorithms or postulations in the context of ontology evolution could also prove interesting and uncover useful approaches to this problem. The semantically different operations of update and erasure [KM90a] are worth studying as well.

The proposed migration of the AGM theory in the context of Tarski's model is not complete either, as only the contraction operator was considered; future work should address the problem of revision as well, research on which is currently at a preliminary stage. As already mentioned, AGM-compliance is a property that simply guarantees the existence of a contraction operator that satisfies the (generalized) AGM postulates in the logic under question. The extent to which the richness of results related to the AGM theory can be applied to AGM-compliant logics (and, in particular, AGM-compliant DLs) still remains to be determined. Furthermore, the connection of AGM-compliance with the AGM-related representation results [Gar92a] is still unexplored.

References

[AADW05] A. Analyti, G. Antoniou, C.V. Damasio, G. Wagner. Stable Model Theory for Extended RDF Ontologies. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 21-36, 2005.

[AFMWZ01] A. Artale, E. Franconi, M. Mosurovic, F. Wolter, M. Zakharyaschev. The DLR_{US} Temporal Description Logic. In Proceedings of the 2001 Description Logic Workshop (DL-01), pages 96-105, 2001.

[AGM85] C. Alchourron, P. Gärdenfors, D. Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. Journal of Symbolic Logic, 50:510-530, 1985.

[AGY05] P. Avesani, F. Giunchiglia, M. Yatskevich. A Large Scale Taxonomy Mapping Evaluation. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 67-81, 2005.

[AM85] C. Alchourron, D. Makinson. On the Logic of Theory Change: Safe Contraction. Studia Logica 44:405-422, 1985.

[AM86] C. Alchourron, D. Makinson. Maps Between Some Different Kinds of Contraction Function: the Finite Case. Studia Logica 45:187-198, 1986.

[Ant97] G. Antoniou. Nonmonotonic Reasoning. The MIT Press, 1997.

[BAGM99] D. Billington, G. Antoniou, G. Governatori, M. Maher. Revising Nonmonotonic Theories: The Case of Defeasible Logic. In Proceedings of the 22nd German Conference on Artificial Intelligence (KI-99), Lecture Notes in Artificial Intelligence (LNAI), Volume 1701, Springer-Verlag, 1999.

[BBH96] F. Baader, M. Buchheit, B. Hollunder. Cardinality Restrictions on Concepts. Artificial Intelligence, 88(1–2):195–213, 1996.

[BCM99] T. Bench-Capon, G. Malcolm. Formalizing Ontologies and Their Relations. In Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA-99), pages 250-259, 1999.

[BCMNPS02] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (eds). The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2002.

[BHS03] F. Baader, I. Horrocks, U. Sattler. Description Logics as Ontology Languages for the Semantic Web. In D. Hutter, W. Stephan (eds). Festschrift in honor of

Jörg Siekmann, Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, 2003.

[BKBW04] S. Benferhat, S. Kaci, D. Le Berre, M. Williams. Weakening Conflicting Information for Iterated Revision and Knowledge Integration. Artificial Intelligence, 153:339-371, 2004.

[BKKK87] J. Banerjee, W. Kim, H.J. Kim, H.F. Korth. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. In Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, pages 311-322, 1987.

[BLHL01] T. Berners-Lee, J. Hendler, O. Lassila. The Semantic Web. Scientific American, 284(5):34-43, 2001.

[BMRME04] J. de Bruijn, F. Martin-Recuerda, D. Manov, M. Ehrig. D4.2.1: State of the Art Survey on Ontology Merging and Aligning. 2004. Available on the Web (last visited February, 2006): http://www.aifb.uni-karlsruhe.de/WBS/meh/publications/debruijn04state.pdf

[Boch02a] A. Bochman. Entrenchment Versus Dependence: Coherence and Foundations in Belief Change. Journal of Logic, Language and Computation, 11(1):3-27, 2002.

[Boch02b] A. Bochman. Two Representations for Iterative Non-prioritized Change. In Proceedings of the 9th International Workshop on Non-Monotonic Reasoning (NMR-02), 2002.

[Bor96] A. Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. Artificial Intelligence, 82, pages 353-367, 1996.

[BS01] F. Baader, U. Sattler. An Overview of Tableau Algorithms for Description Logics. Studia Logica, 69, pages 5-40, 2001.

[BS81] S.N. Burris, H.P. Sankappanavar. A Course in Universal Algebra. Free Millenium Edition of the original 1981 edition by Springer-Verlag. Available on the Web (last visited February, 2006): http://www.math.uwaterloo.ca/~snburris/htdocs/ualg.html

[Bur98] S.N. Burris. Logic For Mathematics And Computer Science. Prentice Hall, New Jersey, 1998.

[Cal96] D. Calvanese. Reasoning with Inclusion Axioms in Description Logics: Algorithms and Complexity. In Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96), pages 303-307, 1996.

[Cha00] H. Chalupsky. OntoMorph: A Translation System for Symbolic Knowledge. In Proceedings of the 7th International Conference on Knowledge Representation and Reasoning (KR-00), 2000.

[Dal88a] M. Dalal. Investigations Into a Theory of Knowledge Base Revision: Preliminary Report. In Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88), pages 475-479, 1988.

[Dal88b] M. Dalal. Updates in Propositional Databases. Technical Report, DCS-TR-222, Department of Computer Science, Rutgers University, 1988.

[Del01] J.P. Delgrande. A Minimal Modelling for Successful Knowledge Base Revision. In Frontiers of Belief Revision, M.A. Williams, H. Rott (eds). Applied Logic Series, Volume 22, Series Editors: D. Gabbay, J. Barwise, Kluwer Academic Publishers, 2001.

[Del03] J.P. Delgrande. A Consistency-Based Approach for Belief Change. Artificial Intelligence, 151(1-2):1-41, 2003.

[DLNS96] F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf . Reasoning in Description Logics. In G. Brewka (ed). Foundations of Knowledge Representation, pages 191-236. CSLI-Publications, 1996.

[DNR97] F.M. Donini, D. Nardi, R. Rosati. Autoepistemic Description Logics. In Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), pages 136-141, 1997.

[Doy79] J. Doyle. A Truth Maintenance System. Artificial Intelligence, 12(3):231-272, 1979.

[Doy91] J. Doyle. Rational Belief Revision (Preliminary Report). In Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR-91), pages 163-174, 1991.

[DP97] A. Darwiche, J. Pearl. On the Logic of Iterated Belief Revision. Artificial Intelligence, 89(1-2):1-29, 1997.

[DSBHH+04] M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuiness, P. Patel-Schneider, L.A. Stein. OWL Web Ontology Language Reference. W3C Recommendation, 2004. Available on the Web (last visited February, 2006): http://www.w3.org/TR/owl-ref/

[DSWKB00] A.J. Duineveld, R. Stoter, M.R. Weiden, B. Kenepa, V.R. Benjamins. WonderTools? A Comparative Study of Ontological Engineering Tools. International Journal of Human-Computer Studies, 52(6):1111-1133, 2000.

[EBBBB+04] J. Euzenat, T. Le Bach, J. Barrasa, P. Bouquet, J. de Bo, R. Dieng, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, D. Maynard, A. Napoli, G. Stamou, H. Stuckeschmidt, P. Shvaiko, S. Tessaris, S. van Acker, I. Zaihrayeu. D2.2.3: State of the Art on Ontology Alignment. 2004. Available on the Web (last visited February, 2006): http://www.starlab.vub.ac.be/research/projects/knowledgeweb/kweb-223.pdf

[End72] H.B. Enderton. A Mathematical Introduction to Logic. Academic Press, New York, 1972.

[End77] H.B. Enderton. Elements of Set Theory. Academic Press, New York, 1977.

[ESS05] M. Ehrig, S. Staab, Y. Sure. Bootstrapping Ontology Alignment Methods with APFEL. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 186-200, 2005.

[Fer04] A. Ferrara. Methods and Techniques for Ontology Matching and Evolution in Open Distributed Systems. In Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAISE-04), 2004.

[FGM00] E. Franconi, F. Grandi, F. Mandreoli. A Semantic Approach for Schema Evolution and Versioning in Object-Oriented Databases. In Proceedings of the 6th International Conference on Rules and Objects in Databases (DOOD-00), 2000.

[FH94] A. Fuhrmann, S.O. Hansson. A Survey of Multiple Contractions. Journal of Logic, Language and Information, 3:39-76, 1994.

[Flou04] G. Flouris. Belief Change in Arbitrary Logics. In Proceedings of the 3rd Hellenic Data Management Symposium (HDMS-04), PhD Presentations Session, pages 217-219, 2004.

[Flou05] G. Flouris. Applying Belief Change to Ontology Evolution. In Proceedings of the 1st International Semantic Web Doctoral Symposium (ISWDS-05), Research Thesis Summary, pages 9-12, 2005.

[Foo95] N. Foo. Ontology Revision. In G. Ellis, R. Levinson, W. Rich, J. Sowa (eds). Proceedings of the 3rd International Conference on Conceptual Structures (ICCS-95), Lecture Notes in Artificial Intelligence (LNAI), Volume 954, Springer-Verlag, pages 16-31, 1995.

[FP01a] G. Flouris, D. Plexousakis. Belief Revision in Propositional Knowledge Bases. In Proceedings of the 8th Panhellenic Conference on Informatics, pages 412-421, 2001.

[FP01b] G. Flouris, D. Plexousakis. Belief Revision Using Table Transformation Technical Report FORTH-ICS/TR-290, July 2001.

[FP03] G. Flouris, D. Plexousakis. On the Use of Matrices for Belief Revision. In Y. Manolopoulos, S. Evripidou, A. Kakas (eds). Vol. 2563, Advances in Informatics, Post-proceedings of the 8th Panhellenic Conference in Informatics, 2003.

[FP05] G. Flouris, D. Plexousakis. Handling Ontology Change: Survey and Proposal for a Future Research Direction. Technical Report FORTH-ICS/TR-362, September 2005.

[FP06] G. Flouris, D. Plexousakis. Bridging Ontology Evolution and Belief Change. In Proceedings of the 4th Hellenic Conference on Artificial Intelligence (SETN-06), 2006.

[FPA04a] G. Flouris, D. Plexousakis, G. Antoniou. AGM Postulates in Arbitrary Logics: Initial Results and Applications. Technical Report FORTH-ICS/TR-336, April 2004.

[FPA04b] G. Flouris, D. Plexousakis, G. Antoniou. Generalizing the AGM Postulates: Preliminary Results and Applications. In Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR-04), pages 171-179, 2004.

[FPA05a] G. Flouris, D. Plexousakis, G. Antoniou. On Applying the AGM Theory to DLs and OWL. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 216-231, 2005.

[FPA05b] G. Flouris, D. Plexousakis, G. Antoniou. Ontology Evolution Using Belief Change and the AGM Theory. In the Poster Session of the 4th International Semantic Web Conference (ISWC-05), A0-sized poster, 2005.

[FPA05c] G. Flouris, D. Plexousakis, G. Antoniou. Updating Description Logics Using the AGM Theory. In Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning, pages 69-76, 2005.

[FPA05d] G. Flouris, D. Plexousakis, G. Antoniou. Updating DLs Using the AGM Theory: A Preliminary Study. In Proceedings of the Poster Session of the 2005 Description Logic Workshop (DL-05), 2005.

[FPA06] G. Flouris, D. Plexousakis, G. Antoniou. Evolving Ontology Evolution. In Proceedings of the 32nd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM-06), Invited Talk, 2006.

[FS05] G. Friedrich, K. Shchekotykhin. A General Diagnosis Method for Ontologies. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 232-246, 2005.

[Fuhr91] A. Fuhrmann. Theory Contraction Through Base Contraction. Journal of Philosophical Logic, 20:175-203, 1991.

[FUV83] R. Fagin, J.D. Ullman, M.Y. Vardi. On the Semantics of Updates in Databases. In Proceedings of the 2nd SIGACT-SIGMOD Symposium on Principles of Database Systems, pages 352-365, 1983.

[Gar92a] P. Gärdenfors. Belief Revision: An Introduction. In P. Gärdenfors (ed). Belief Revision, pages 1-20, Cambridge University Press, 1992.

[Gar92b] P. Gärdenfors. The Dynamics of Belief Systems: Foundations Versus Coherence Theories. Revue Internationale de Philosophie 44, pages 24-46. Reprinted in C. Bicchieri, M. L. Dalla Chiara (eds). Knowledge, Belief and Strategic Interaction, pages 377-396, Cambridge University Press, 1992.

[GHVD03] B.N. Grosof, I. Horrocks, R. Volz, S. Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In Proceedings of the 12th International World Wide Web Conference (WWW-03), 2003.

[GM88] P. Gärdenfors, D. Makinson. Revisions of Knowledge Systems Using Epistemic Entrenchment. In Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning About Knowledge (TARK-88), pages 83-95, 1988.

[GN94] J. Giesl, I. Neumann. The Semantics of Rational Contractions. Technical Report 29/94, Universität Karlsruhe, Germany, 1994.

[Gra71] G. Gratzer. Lattice Theory: First Concepts And Distributive Lattices. W. H. Freeman and co. San Francisco, 1971.

[Gro88] A. Grove. Two Modellings for Theory Change. Journal of Philosophical Logic, 17:157-170, 1988.

[Gru93] T.R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2):199-220, 1993.

[Hal74] P. Halmos. Naïve Set Theory. Springer-Verlag, 1974.

[Han92] S.O. Hansson. In Defense of Base Contraction. Synthese 91:239-245, 1992.

[Han93] S.O. Hansson. Theory Contraction and Base Contraction Unified. Journal of Symbolic Logic, 58(2):602-625, 1993.

[Han96] S.O. Hansson. Knowledge-level Analysis of Belief Base Operations. Artificial Intelligence, 82:215-235, 1996.

[Han98] S.O. Hansson. Revision of Belief Sets and Belief Bases. Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 3, pages 17-75, 1998.

[Han99a] S.O. Hansson. A Survey of Non-Prioritized Belief Revision. Erkenntnis, 50:413-427, 1999.

[Han99b] S. O. Hansson. A Textbook Of Belief Dynamics. Kluwer Academic Publishers, 1999.

[HD05] A. Hunter, J. Delgrande. Iterated Belief Change: A Transition System Approach. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), 2005.

[HH00] J. Heflin, J. Hendler. Dynamic Ontologies on the Web. In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00), pages 443-449, 2000.

[HHHSS05] P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, Y. Sure. A Framework for Handling Inconsistency in Changing Ontologies. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 353-367, 2005.

[HHL99] J. Heflin, J. Hendler, S. Luke. Coping with Changing Ontologies in a Distributed Environment. In Proceedings of the Workshop on Ontology Management of the 16th National Conference on Artificial Intelligence (AAAI-99), WS-99-13, AAAI Press, pages 74-79, 1999.

[HHPSBL+01] I. Horrocks, F. van Harmelen, P. Patel-Schneider, T. Berners-Lee, D. Brickley, D. Connolly, M. Dean, S. Decker, D. Fensel, R. Fikes, P. Hayes, J. Heflin, J. Hendler, O. Lassila, D. McGuinness, L.A. Stein. DAML+OIL, 2001. Available on the Web (last visited February, 2006): http://www.daml.org/2001/03/daml+oil-index.html

[HHTGV04] Z. Huang, F. van Harmelen, A. ten Teije, P. Groot, G. Visser. Reasoning with Inconsistent Ontologies: Framework and Prototype. SEKT Deliverable D3.4.1, 2004. Available on the Web (last visited February, 2006): http://wasp.cs.vu.nl/~huang/papers/sekt341.pdf

[HHTGV05] Z. Huang, F. van Harmelen, A. ten Teije, P. Groot, G. Visser. Reasoning with Inconsistent Ontologies. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), 2005.

[HPS03] I. Horrocks, P. Patel-Schneider. Three Theses of Representation in the Semantic Web. In Proceedings of the 12th International World Wide Web Conference (WWW-03), pages 39-47, 2003.

[HPS04] I. Horrocks, P. Patel-Schneider. Reducing OWL Entailment to Description Logic Satisfiability. Journal of Web Semantics, 1(4):345-357, 2004.

[HPSH03] I. Horrocks, P. Patel-Schneider, F. van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. Journal of Web Semantics, 1(1):7-26, 2003.

[HS00] U. Hustadt, R.A. Schmidt. Issues of Decidability for Description Logics in the Framework of Resolution. In Automated Deduction in Classical and Non-Classical Logics, Lecture Notes in Artificial Intelligence (LNAI), Volume 1761, pages 191-205, Springer-Verlag, 2000.

[HS01] I. Horrocks, U. Sattler. Ontology Reasoning in the SHOQ(D) Description Logic. In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), pages 199-204, 2001.

[HS04] P. Haase, Y. Sure. D3.1.1.b State of the Art on Ontology Evolution. 2004. Available on the Web (last visited February, 2006): http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/SEKT-D3.1.1.b.pdf

[HS05a] P. Haase, L. Stojanovic. Consistent Evolution of OWL Ontologies. In Proceedings of the 2nd European Semantic Web Conference (ESWC-05), 2005.

[HS05b] Z. Huang, H. Stuckenschmidt. Reasoning with Multi-version Ontologies: A Temporal Logic Approach. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 398-412, 2005.

[HST99] I. Horrocks, U. Sattler, S. Tobies. Practical Reasoning for Very Expressive Description Logics. In H. Ganzinger, D. McAllester, A. Voronkov (eds). Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR-99), Lecture Notes in Artificial Intelligence (LNAI), Volume 1705, pages 161-180, Springer-Verlag, 1999.

[KC88] W. Kim, H.T. Chou. Versions of Schema for Object-Oriented Databases. In Proceedings of the 14th International Conference on Very Large Data Bases (VLDB-88), pages 148-159, 1988.

[Kel98] K. Kelly. The Learning Power of Belief Revision. In Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-98), pages 111-124, 1998.

[KF01] M. Klein, D. Fensel. Ontology Versioning on the Semantic Web. In Proceedings of the International Semantic Web Working Symposium (SWWS), pages 75-91, 2001.

[KFKO02] M. Klein, D. Fensel, A. Kiryakov, D. Ognyanov. Ontology Versioning and Change Detection on the Web. In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW-02), 2002.

[KL04] S.H. Kang, S.K. Lau. Ontology Revision Using the Concept of Belief Revision. In Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES-04), part III, pages 8-15, 2004.

[KLM05] S. Konieczny, J. Lang, P. Marquis. Reasoning Under Inconsistency: The Forgotten Connective. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), 2005.

[KM90a] H. Katsuno, A.O. Mendelzon. On the Difference Between Updating a Knowledge Base and Revising It. Technical Report on Knowledge Representation and Reasoning, University of Toronto, Canada, KRR-TR-90-6, 1990.

[KM90b] H. Katsuno, A.O. Mendelzon. Propositional Knowledge Base Revision and Minimal Change. Technical Report on Knowledge Representation and Reasoning, University of Toronto, Canada, KRR-TR-90-3, 1990.

[KN03] M. Klein, N.F. Noy. A Component-Based Framework for Ontology Evolution. In Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems, CEUR-WS, vol. 71, 2003.

[Kon00] S. Konieczny. On the Difference Between Merging Knowledge Bases and Combining Them. In Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-00), pages 135-144, 2000.

[KPP05] S. Konieczny, R. Pino-Perez. Propositional Belief Base Merging or How to Merge Beliefs/Goals Coming from Several Sources and Some Links with Social Choice Theory. European Journal of Operational Research, 160(3):785-802, 2005.

[KPS05] A. Kalyanpur, B. Parsia, E. Sirin. Black Box Debugging of Unsatisfiable Classes. In the Poster Session of the 4th International Semantic Web Conference (ISWC-05), PID-90, 2005.

[KS03] Y. Kalfoglou, M. Schorlemmer. Ontology Mapping: the State of the Art. Knowledge Engineering Review, 18(1), pages 1-31, 2003.

[LE03] P. Lambrix, A. Edberg. Evaluation of Ontology Merging Tools in Bioinformatics. In Proceedings of the 8th Pacific Symposium on Biocomputing, pages 589-600, 2003.

[Lib97] P. Liberatore. The Complexity of Iterated Belief Revision. In Proceedings of the 6th International Conference on Database Theory (ICDT-97), pages 276-290, 1997.

[LM04] K. Lee, T. Meyer. A Classification of Ontology Modification. In Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AI-04), pages 248-258, 2004.

[LS00] C. Lutz, U. Sattler. Mary Likes All Cats. In Proceedings of the 2000 International Workshop in Description Logics (DL-00), pages 213-226, 2000.

[LSRH97] S. Luke, L. Spector, D. Rager, J. Hendler. Ontology-based Web Agents. In Proceedings of the 1st International Conference on Autonomous Agents, pages 59-66, 1997.

[LSV05] S.C. Lam, D.H. Sleeman, W. Vasconselos. ReTAX++: A Tool for Browsing and Revising Ontologies. In Poster Proceedings of the 4th International Semantic Web Conference (ISWC-05), PID-33, 2005.

[Mak85] D. Makinson. How to Give It Up: A Survey of Some Formal Aspects of the Logic of Theory Change. Synthese 62, pages 347-363, 1985.

[Mak87] D. Makinson. On the Status of the Postulate of Recovery in the Logic of Theory Change. Journal of Philosophical Logic 16:383-394, 1987.

[Mak93] D. Makinson. Five Faces of Minimality. Studia Logica, 52:339-379, 1993.

[MBJK90] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis. Telos: Representing Knowledge About Information Systems. ACM Transactions on Information Systems, 8(4):325-362, 1990.

[Men87] E. Mendelson. Introduction to Mathematical Logic. Wadsworth and Brooks/Cole Advanced Books and Software, Pacific Grove, California, 1987.

[MFRW00] D. McGuiness, R. Fikes, J. Rice, S. Wilder. An Environment for Merging and Testing Large Ontologies. In Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-00), Technical Report KSL-00-16, Knowledge Systems Laboratory, Stanford University, 2000.

[Mil95] G.A. Miller. WordNet: A Lexical Database for English. Communications of the ACM (CACM), 38(11):39-41, 1995.

[MLB05] T. Meyer, K. Lee, R. Booth. Knowledge Integration for Description Logics. In Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning, 2005.

[MMSRV03] A. Maedche, B. Motik, L. Stojanovic, R. Studer, R. Volz. An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies. In Proceedings of the 12th International World Wide Web Conference (WWW-03), 2003.

[MNJ05] P. Mitra, N.F. Noy, A.R. Jaiswal. OMEN: A Probabilistic Ontology Mapping Tool. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 537-547, 2005.

[MSB06] E. Miller, R. Swick, D. Brickley. Resource Description Framework (RDF) / W3C Semantic Web Activity, 2006. Available on the Web (last visited February, 2006):

http://www.w3.org/RDF/

[MSCK05] M. Magiridou, S. Sahtouris, V. Christophides, M. Koubarakis. RUL: A Declarative Update Language for RDF. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 506-521, 2005.

[Nay94] A.C. Nayak. Iterated Belief Change Based on Epistemic Entrenchment. Erkenntnis, 41:353-390, 1994.

[Neb89] B. Nebel. A Knowledge Level Analysis of Belief Revision. In Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR-89), pages 301-311, 1989.

[New82] A. Newell. The Knowledge Level. Artificial Intelligence, 18(1), 1982.

[NK04] N.F. Noy, M. Klein. Ontology Evolution: Not the Same as Schema Evolution. Knowledge and Information Systems, 6(4), pages 428-440. Available as SMI technical report SMI-2002-0926, 2004.

[NKKM04] N.F. Noy, S. Kunnatur, M. Klein, M.A. Musen. Tracking Changes During Ontology Evolution. In Proceedings of the 3rd International Conference on the Semantic Web (ISWC-04), published as Lecture Notes in Computer Science (LNCS), Volume 3298, pages 259-273, S.A. McIlraith, D. Plexousakis, F. van Harmelen (eds), Springer-Verlag, 2004.

[NM99a] N.F. Noy, M.A. Musen. An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support. In Proceedings of the Workshop on Ontology Management at 16th National Conference on Artificial Intelligence (AAAI-99). Available as SMI technical report SMI-1999-0799, 1999.

[NM99b] N.F. Noy, M.A. Musen. SMART: Automated Support for Ontology Merging and Alignment. In Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management. Available as SMI technical report SMI-1999-0813, 1999.

[NM00] N.F. Noy, M.A. Musen. Algorithm and Tool for Automated Ontology Merging and Alignment. In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831, 2000.

[PDYP05] R. Pan, Z. Ding, Y. Yu, Y. Peng. A Bayesian Network Approach to Ontology Mapping. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 563-577, 2005.

[PGPM99] H.S. Pinto, A. Gomez-Perez, J.P. Martins. Some Issues on Ontology Integration. In Proceedings of the Workshop on Ontologies and Problem-Solving Methods (KRR5) at 16th International Joint Conference on Artificial Intelligence (IJCAI-99), 1999.

[PO97] R.J. Peters, T. Ozsu. An Axiomatic Model of Dynamic Schema Evolution in Objectbase Systems. ACM Transactions on Database Systems, 22(1):75-114, 1997.

[PT05] P. Plessers, O. De Troyer. Ontology Change Detection Using a Version Log. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 578-592, 2005.

[RR85] H. Rubin, J.E. Rubin. Equivalents of the Axiom of Choice. North-Holland, Amsterdam, 1985.

[SC03] S. Schlobach, R. Cornet. Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), 2003.

[SK03] H. Stuckenschmidt, M. Klein. Integrity and Change in Modular Ontologies. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), 2003.

[SM02] G. Stumme, A. Maedche. Ontology Merging for Federated Ontologies on the Semantic Web. In E. Franconi, K. Barker, D. Calvanese (eds). Proceedings of the International Workshop on Foundations of Models for Information Integration (FMII-01), Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, 2002.

[SMMS02] L. Stojanovic, A. Maedche, B. Motik, N. Stojanovic. User-driven Ontology Evolution Management. In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW-02), published as Lecture Notes in Computer Science (LNCS), Volume 2473, pages 285-300, Springer-Verlag, 2002.

[SMSS03] L. Stojanovic, A. Maedche, N. Stojanovic, R. Studer. Ontology Evolution as Reconfiguration-Design Problem Solving. In Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP-03), pages 162-171, 2003.

[Sosa80] E. Sosa. The Raft and the Pyramid: Coherence Versus Foundations in the Theory of Knowledge. Midwest Studies in Philosophy, 5:3-25, 1980.

[SSK05] G. Stoilos, G. Stamou, S. Kollias. A String Metric for Ontology Alignment. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 624-637, 2005.

[SSS91] M. Schmidt-Schaub, G. Smolka. Attributive Concept Descriptions with Complements. Artificial Intelligence, 48:1-26, 1991.

[TCGJSS93] A.U. Tansel, J. Clifford, S.K. Gadia, S. Jajodia, A. Segev, R.T. Snodgrass. Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummings, 1993.

[TXZZZY05] K.W. Tu, M. Xiong, L. Zhang, H.P. Zhu, J. Zhang, Y. Yu. Towards Imaging Large-Scale Ontologies for Quick Understanding and Analysis. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 702-715, 2005.

[Wass98] R. Wassermann. Revising Concepts. In Proceedings of the 5th Workshop on Logic, Language, Information and Communication (WoLLIC-98), 1998.

[Web86] A. Weber. Updating Propositional Formulas. In Proceedings of the 1st International Conference on Expert Database Systems (EDS-86), pages 487-500, 1986.

[WF99] R. Wassermann, E. Fermé. A Note on Prototype Revision. In Spinning Ideas (Electronic Essays dedicated to Peter Gärdenfors on his 50th Birthday). Also available as Research Report PP-1999-18 (ILLC Series), 1999.

[WHRDS05] H. Wang, M. Horridge, A. Rector, N. Drummond, J. Seidenberg. Debugging OWL-DL Ontologies: A Heuristic Approach. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 745-757, 2005.

[Wiki06] Wikipedia Foundation, Inc. Wikipedia, the Free Encyclopedia. Web Page (last visited February, 2006):
http://en.wikipedia.org/wiki/Main_Page

[Wil97] M.A. Williams. Anytime Belief Revision. In Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), 1997.

[Wil98] M.A. Williams. Applications of Belief Revision. In Transactions and Change in Logic Databases, Lecture Notes in Artificial Intelligence (LNAI), Volume 1472, Springer-Verlag, pages 285-314, 1998.

[Wit02] R. Witte. Fuzzy Belief Revision. In Proceedings of the 9th International Workshop on Non-Monotonic Reasoning (NMR-02), 2002.

[Zad65] L.A. Zadeh. Fuzzy Sets. In Information and Control, Vol. 8, pages 338-353, 1965.

[Zol06] E. Zolin. Navigator on Description Logics Complexity. Web Page (last visited February, 2006):
http://www.cs.man.ac.uk/~ezolin/logic/complexity.html

[ZZLZY03] Z. Zhang, L. Zhang, C.X. Lin, Y. Zhao, Y. Yu. Data Migration for Ontology Evolution. In Poster Proceedings of the 2nd International Semantic Web Conference (ISWC-03), 2003.