
**NeOn: Lifecycle Support for Networked Ontologies**

**Integrated Project (IST-2005-027595)**

**Priority: IST-2004-2.4.7 — "Semantic-based knowledge and content systems"**

# D1.2.1 Consistency Models for Networked Ontologies

**Deliverable Co-ordinator:** **Guilin Qi, Peter Haase, Qiu Ji**

**Deliverable Co-ordinating Institution:** **University of Karlsruhe**

An important aspect in the evolution of networked ontologies is to ensure the consistency of the ontologies. In this deliverable we therefore address the question of how to define appropriate models for consistency in such evolving networks of ontologies. We first provide a comparison of existing approaches to dealing with inconsistencies in evolving ontologies. We then propose a novel general approach for resolving inconsistency and incoherence in ontologies. Our approach differs from current approaches mainly in two aspects. First, when revising an ontology, our approach deals with logical inconsistency and logical incoherence in an integrated way. Second, we classify logical inconsistency into three categories and handle different kinds of inconsistency using different revision strategies. We instantiate our approach by proposing concrete approaches to resolving incoherence and inconsistency. Finally, we provide implementations of the proposed methods.

| | | | |
|---|---|---|---|
| Document Identifier: | NEON/2007/D1.2.1/v1.0 | Date due: | February 28, 2007 |
| Class Deliverable: | NEON EU-IST-2005-027595 | Submission date: | March 30, 2007 |
| Project start date | March 1, 2006 | Version: | v1.0 |
| Project duration: | 4 years | State: | Final |
| | | Distribution: | Public |

## NeOn Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities, grant number IST-2005-027595. The following partners are involved in the project:


| | |
|---|---|
| **Open University (OU) – Coordinator**<br>Knowledge Media Institute – KMi<br>Berrill Building, Walton Hall<br>Milton Keynes, MK7 6AA<br>United Kingdom<br>Contact person: Martin Dzbor, Enrico Motta<br>E-mail address: {m.dzbor, e.motta}@open.ac.uk | **Universität Karlsruhe – TH (UKARL)**<br>Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB<br>D-76128 Karlsruhe<br>Germany<br>Contact person: Peter Haase<br>E-mail address: pha@aifb.uni-karlsruhe.de |
| **Universidad Politécnica de Madrid (UPM)**<br>Campus de Montegancedo<br>28660 Boadilla del Monte<br>Spain<br>Contact person: Asunción Gómez Pérez<br>E-mail address: asun@fi.ump.es | **Software AG (SAG)**<br>Uhlandstrasse 12<br>64297 Darmstadt<br>Germany<br>Contact person: Walter Waterfeld<br>E-mail address: walter.waterfeld@softwareag.com |
| **Intelligent Software Components S.A. (ISOCO)**<br>Calle de Pedro de Valdivia 10<br>28006 Madrid<br>Spain<br>Contact person: Richard Benjamins<br>E-mail adress: rbenjamins@isoco.com | **Institut 'Jožef Stefan' (JSI)**<br>Jamova 39<br>SL–1000 Ljubljana<br>Slovenia<br>Contact person: Marko Grobelnik<br>E-mail address: marko.grobelnik@ijs.si |
| **Institut National de Recherche en Informatique et en Automatique (INRIA)**<br>ZIRST – 665 avenue de l'Europe<br>Montbonnot Saint Martin<br>38334 Saint-Ismier<br>France<br>Contact person: Jérôme Euzenat | **University of Sheffield (USFD)**<br>Dept. of Computer Science<br>Regent Court<br>211 Portobello street<br>S14DP Sheffield<br>United Kingdom<br>Contact person: Hamish Cunningham |
| **Universität Kolenz-Landau (UKO-LD)**<br>Universitätsstrasse 1<br>56070 Koblenz<br>Germany<br>Contact person: Steffen Staab<br>E-mail address: staab@uni-koblenz.de | **Consiglio Nazionale delle Ricerche (CNR)**<br>Institute of cognitive sciences and technologies<br>Via S. Marino della Battaglia<br>44 – 00185 Roma-Lazio Italy<br>Contact person: Aldo Gangemi<br>E-mail address: aldo.gangemi@istc.cnr.it |
| **Ontoprise GmbH. (ONTO)**<br>Amalienbadstr. 36<br>(Raumfabrik 29)<br>76227 Karlsruhe<br>Germany<br>Contact person: Jürgen Angele<br>E-mail address: angele@ontoprise.de | **Asociación Española de Comercio Electrónico (AECE)**<br>C/Icalde Barnils, Avenida Diagonal 437<br>08036 Barcelona<br>Spain<br>Contact person: Jose Luis Zimmerman<br>E-mail address: jlzimmerman@fecemd.org |
| **Food and Agriculture Organization of the United Nations (FAO)**<br>Viale delle Terme di Caracalla<br>00100 Rome, Italy<br>Contact person: Marta Iglesias<br>E-mail address: marta.iglesias@fao.org | **Atos Origin S.A. (ATOS)**<br>Calle de Albarracín, 25<br>28037 Madrid<br>Spain<br>Contact person: Tomás Pariente Lobo<br>E-mail address: tomas.parientelobo@atosorigin.com |

## Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed writing parts of this document:

- University of Karlsruhe

# Executive Summary

Next generation semantic applications will be characterized by a large number of ontologies, some of them constantly evolving. This new generation of applications relies on ontologies embedded in a network of already existing ontologies. An important aspect in the evolution of these networked ontologies is to ensure the consistency of the ontologies. We argue that in this scenario it will be inadequate to maintain single, globally consistent semantic model that serves the needs of application developers and fully integrates a number of pre-existing ontologies. In this deliverable we therefore address the question of how to define appropriate models for consistency in such evolving networks of ontologies.

We first provide a comparison of existing approaches to dealing with inconsistencies in evolving ontologies. In the comparison, we evaluate how these approaches are applicable to networked scenarios.

We then propose a novel general approach for resolving inconsistency and incoherence in ontologies. Our approach differs from current approaches mainly in two aspects. First, when revising an ontology, our approach deals with logical inconsistency and logical incoherence in an integrated way. Second, we classify logical inconsistency into three categories and handle different kinds of inconsistency using different revision strategies. Our general approach is independent of specific approaches to resolving with inconsistency or incoherence. We instantiate our approach by proposing concrete approaches to resolving incoherence and inconsistency.

Finally, we provide implementations of the proposed methods. The implementations are available in two forms: (1) We have developed the RaDON system for *Reasoning and Diagnosis in Ontology Networks*, which makes the proposed functionalities accessible on top of the KAON2 reasoner via extensions to the DIG interface. (2) We have integrated the new functionalities into the KAON2 OWL Tools, which allow to access the functionalities from the command line.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 The NeOn Big Picture

Next generation semantic applications will be characterized by a large number of ontologies, some of them constantly evolving. As the complexity of semantic applications increases, more and more knowledge will be embedded in applications, typically drawn from a wide variety of sources. This new generation of applications will thus likely rely on ontologies embedded in a network of already existing ontologies. Ontologies and metadata will have to be kept up to date when application environments and users' needs change. We argue that in this scenario it will become prohibitively expensive for people to directly adopt the current approach to semantic integration, where the expectation is to produce a single, globally consistent semantic model that serves the needs of application developers and fully integrates a number of pre-existing ontologies. In contrast to the current model, future applications will very likely rely on networks of contextualized ontologies, which are usually locally, but not globally consistent.

This report is part of the work performed in WP 1 on "Dynamics of Networked Ontologies". The goal of this work package is to develop an integrated approach for the evolution process of networked ontologies and related metadata. As shown in Figure 1.1, WP1 belongs to the central part of the research and development WPs in NeOn. The tasks of WP1 are heavily inter-related with other work packages. For the individual phases of the process we will develop new methods that consider the complex relationships in a network of ontologies. These include dependencies, mappings, different versions and also take possible inconsistencies into account.

Specific goals in this workpackage include support for:

1. representing, managing and interpreting dependencies between multiple networked ontologies
2. evolution of networked ontologies in exploiting various models of change propagation, which have different applicabilities depending on the model of coordination and control
3. maintaining partial/local consistency of a set of networked ontologies, which might not be globally consistent
4. evolving metadata along with changing ontologies and predicting future structural changes in ontologies.

## 1.2 Motivation

Ontologies play a crucial role for the success of the Semantic Web [BLHL01]. There are many representation languages for ontologies, such as description logics (or DLs for short) and F-logic [SS04]. Recently, the problem of inconsistency (or incoherence) handling in ontologies has attracted a lot of attention. Inconsistency can occur due to several reasons, such as modelling errors, migration or merging ontologies, and

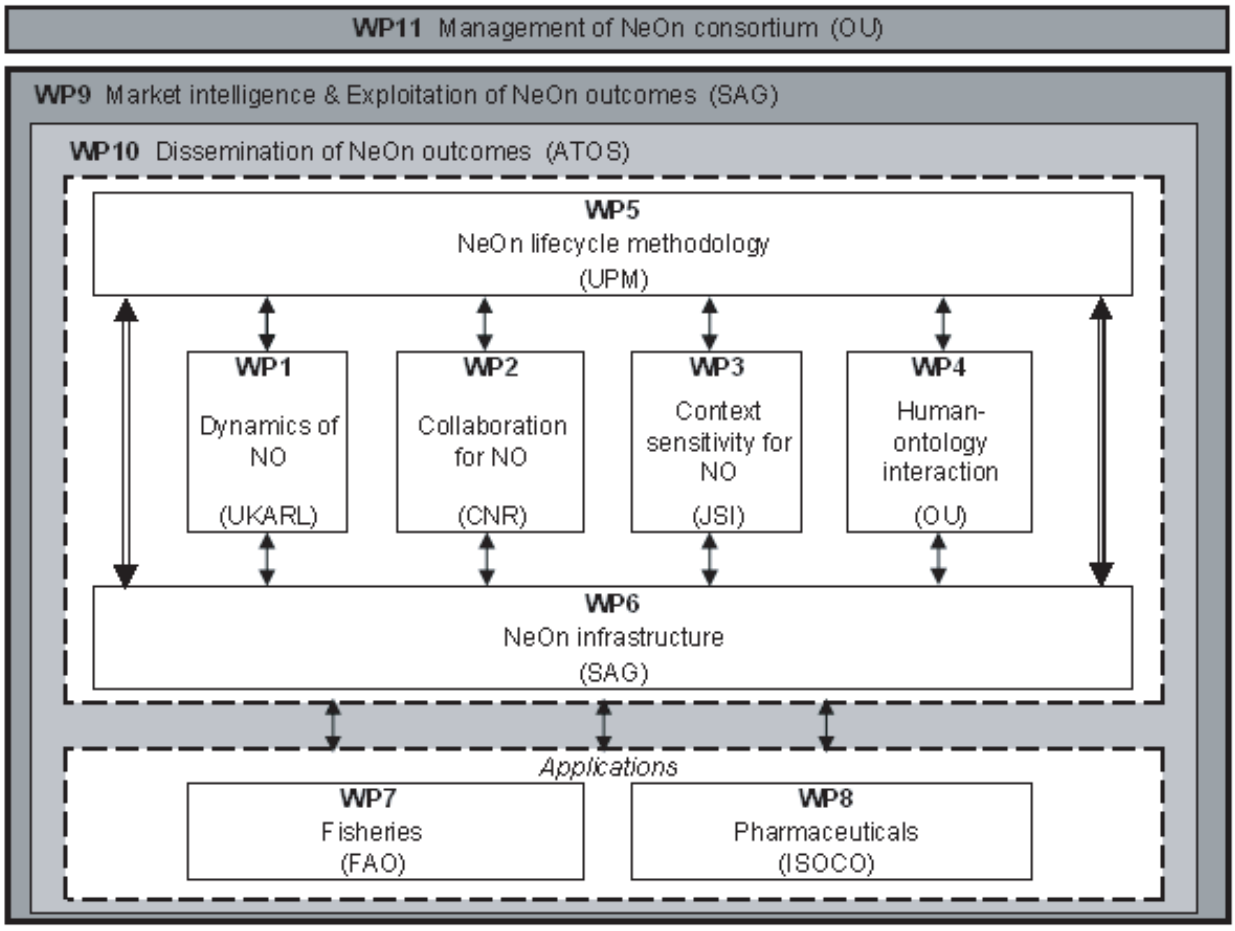


Figure 1.1: Relationships between different workpackages in NeOn

ontology evolution. Current DL reasoners, such as RACER [HM01] and FaCT [Hor98], can detect logical inconsistency. However, they only provide lists of unsatisfiable classes. The process of *resolving* inconsistency is left to the user or ontology engineers. The need to improve DL reasoners to reason with inconsistency is becoming urgent to make them more applicable. Many approaches were proposed to handle inconsistency in ontologies based on existing techniques for inconsistency management in traditional logics, such as propositional logic and nonmonotonic logics [PSK05, HvHH$^+$05, Sch05, SC03, FPA05, HvHtT05].

There are mainly two ways to deal with inconsistent ontologies [HvHtT05]. One way is to simply avoid the inconsistency and to apply a non-standard reasoning method to obtain meaningful answers. A general framework for reasoning with inconsistent ontologies based on *concept relevance* was proposed in [HvHtT05]. The idea is to select from an inconsistent ontology some consistent sub-theories based on a *selection function*, which is defined on the syntactic or semantic relevance. Then standard reasoning on the selected sub-theories is applied to find *meaningful* answers. The second way to deal with logical contradictions is to resolve logical modeling errors whenever a logical problem is encountered. In [MLB05], the authors proposed an algorithm for inconsistency handling by transforming every GCI in a DL knowledge base into a cardinality restriction, and a cardinality restriction responsible for a conflict is weakened by relaxing the restrictions on the number of elements it may have. In [QLB06b], a revision-based algorithm for handling inconsistency in description logics is proposed which generalizes the approach in [MLB05]. Several methods have been proposed to debug erroneous terminologies and have them repaired when inconsistencies are detected [SC03, Sch05, PSK05, FS05]. In this deliverable, we consider the second way of resolving inconsistency.

## 1.3 Overview of the Deliverable

This deliverable is structured as follows. In Chapter 2, we provide some basic notions which will be used to define approaches in Chapter 3. In Chapter 3, we first define criteria for the comparison of these approaches and then give an overview of existing approaches. In Chapter 4, we propose some a new general approach for resolving inconsistency and incoherence. We then instantiate our general approach in 5. Further, we discuss implementations of our approaches in Chapter 6. We conclude with a summary and a roadmap for future work in Chapter 7.

# Chapter 2

# Preliminaries

We assume that the reader is familiar with Description Logics (DLs) and refer to Chapter 2 of the DL handbook [BCM+03] for an excellent introduction. A DL knowledge base (or local ontology) $O$ consists of a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. A TBox contains intensional knowledge such as concept definitions of the form $C \sqsubseteq D$, where $C$ and $D$ are concepts. An ABox contains extensional knowledge and is used to describe individuals. Throughout the deliverable, let $\mathcal{T} = \{Ax_1, ..., Ax_n\}$ be a set of (terminological) axioms, where $Ax_i$ is of the form $C_i \sqsubseteq D_i$ for each $1 \leq i \leq n$ and arbitrary concepts $C_i$ and $D_i$. A TBox is called *unfoldable* if the left-hand sides of the axioms (the defined concepts) are atomic, and if the right-hand sides (the definitions) contain no direct or indirect reference to the defined concept [Neb90].
We introduce the notion of incoherence in DLs defined in [FHP+06].

**Definition 1 (Unsatisfiable Concept)** *A concept name $C$ in a terminology $\mathcal{T}$, is unsatisfiable iff, for each model $\mathcal{I}$ of $\mathcal{T}$, $C^{\mathcal{I}} = \emptyset$.*

That would lead us to consider the kinds of terminologies and ontologies with unsatisfiable concepts.

**Definition 2 (Incoherent Terminology)** *A TBox $\mathcal{T}$ is incoherent iff there exists an unsatisfiable concept name in $\mathcal{T}$.*

**Definition 3 (Incoherent Ontology)** *An ontology $O$ is incoherent iff its TBox is incoherent.*

According to the above definitions, we know that the incoherence can occur only in the terminology level. Namely, an ontology $O = \langle \mathcal{T}, \mathcal{A} \rangle$ is incoherent iff its terminology $TBox$ is incoherent. Therefore, when we talked about an ontology, we only mean its TBox. Incoherence does not provide the classical sense of the inconsistency because there might exist a model for an incoherent ontology.

**Definition 4 (Inconsistent Ontology)** *An ontology $O$ is inconsistent iff it has no model.*

However, incoherence and inconsistency related with each other. According to the discussion in [FHP+06], incoherence is potential for the cause of inconsistency. That is, suppose $C$ is an unsatisfiable concept in $\mathcal{T}$, if a concept assertion $C(a)$ exists in the ABox $\mathcal{A}$, then the ontology $O = \langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent.
Current DL reasoners, such as RACER, can detect logical incoherence and return unsatisfiable concepts in OWL ontologies. However, they do not support the diagnosis and incoherence resolution at all. To explain logical incoherence, it is important to debug *relevant* axioms which are responsible for the contradiction.

**Definition 5** *[SC03] Let $A$ be a named concept which is unsatisfiable in a TBox $\mathcal{T}$. A set $\mathcal{T}' \subseteq \mathcal{T}$ is a* minimal unsatisfiability-preserving sub-TBox (MUPS) *of $\mathcal{T}$ if $A$ is unsatisfiable in $\mathcal{T}'$, and $A$ is satisfiable in every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$. The set of all MUPS of $\mathcal{T}$ w.r.t $A$ is denoted as $MU_A(\mathcal{T})$*

A MUPS of $\mathcal{T}$ $w.r.t$ $A$ is the minimal sub-TBox of $\mathcal{T}$ in which $A$ is unsatisfiable. We will abbreviate the set of MUPS of $\mathcal{T}$ $w.r.t$ a concept name $A$ by $mups(\mathcal{T}, A)$. Let us consider an example from [SC03]. Suppose $\mathcal{T}$ contains the following axioms:

$$ax_1 : A_1 \sqsubseteq \neg A \sqcap A_2 \sqcap A_3 \quad ax_2 : A_2 \sqsubseteq A \sqcap A_4$$
$$ax_3 : A_3 \sqsubseteq A_4 \sqcap A_5 \quad ax_4 : A_4 \sqsubseteq \forall s.B \sqcap C$$
$$ax_5 : A_5 \sqsubseteq \exists s.\neg B \quad ax_6 : A_6 \sqsubseteq A_1 \sqcup \exists r.(A_3 \sqcap \neg C \sqcap A_4)$$
$$ax_7 : A_7 \sqsubseteq A_4 \sqcap \exists s.\neg B$$

where $A$, $B$ and $C$ are atomic concept names and $A_i$ ($i = 1, ..., 7$) are defined concept names, and $r$ and $s$ are atomic roles. In this example, the unsatisfiable concept names are $A_1, A_3, A_6, A_7$ and MUPS of $\mathcal{T}$ w.r.t $A_i$ ($i = 1, 3, 6, 7$) are:
$mups(\mathcal{T}, A_1) : \{\{ax_1, ax_2\}, \{ax_1, ax_3, ax_4, ax_5\}\}$
$mups(\mathcal{T}, A_3) : \{ax_3, ax_4, ax_5\}$
$mups(\mathcal{T}, A_6) : \{\{ax_1, ax_2, ax_4, ax_6\}, \{ax_1, ax_3, ax_4, ax_5, ax_6\}\}$
$mups(\mathcal{T}, A_7) : \{ax_4, ax_7\}$
MUPS are useful for relating sets of axioms to the unsatisfiability of specific concepts, but they can also be used to calculate a minimal incoherence preserving sub-TBox, which relates sets of axioms to the incoherence of a TBox in general and is defined as follows.

**Definition 6** *[SC03] Let $\mathcal{T}$ be an incoherent TBox. A TBox $\mathcal{T}' \subseteq \mathcal{T}$ is a* minimal incoherence-preserving sub-TBox (MIPS) *of $\mathcal{T}$ if $\mathcal{T}'$ is incoherent, and every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$ is coherent. The set of all MIPSs of $\mathcal{T}$ is denoted as $MI(\mathcal{T})$.*

A MIPS of $\mathcal{T}$ is the minimal sub-TBox of $\mathcal{T}$ which is incoherent. The set of MIPS for a TBox $\mathcal{T}$ is abbreviated with $mips(\mathcal{T})$. For $\mathcal{T}$ in the above example, we get 3 MIPS:
$mips(\mathcal{T}) = \{\{ax_1, ax_2\}, \{ax_3, ax_4, ax_5\}, \{ax_4, ax_7\}\}$

# Chapter 3

# An Overview of Existing Approaches

In this chapter we provide an overview of existing approaches to resolving inconsistency and incoherence in ontologies. We start with a set of criteria used for the comparison of the approaches.

## 3.1 Criteria

- *Applications:* Resolving inconsistencies is an important issue to address for a number of different tasks in ontology management. While some approaches are general and may be applied for different tasks, other approaches are developed to support particular applications such as *Repair* of inconsistent ontologies, *Evolution* of ontologies, or *Merging* of potentially mutually inconsistent ontologies.
- *Granularity:* Dealing with and resolving inconsistencies can be performed on different levels of granularity. For example, a typical approach is to repair inconsistencies by removing complete axioms. Other approaches are more fine granular in the sense that they allow weaken axioms by changes on to the substructure of the axioms.
- *Preservation of Structure:* Often the algorithms for diagnosing and repairing inconsistencies require some transformation of the knowledge base to some normal form (e.g. negation normal form.) While the thus obtained knowledge bases are logically equivalent, they may be un-intuitive to the user. It is therefore desirable to preserve the structure of the original axioms wherever possible.
- *Inconsistency vs. Incoherence:* Inconsistency is often used as a term to refer to a number of different types of conflicts in a knowledge base. On the level of the TBox, typically the notions of *unsatisfiability* and *incoherence* are relevant. A concept is *unsatisfiable* w.r.t. a terminology if, and only if its interpretation is empty in every model of the terminology. A TBox is incoherent if it contains an unsatisfiable concept. On the other hand, *inconsistency* of an ontology means that there exists no model at all for the ontology. Inconsistency may occur both in the TBox and the ABox.
- *Support for ABox, TBox:* In Description Logics, the problem of diagnosis has classically focused on dealing with coherence on the terminological level. In many applications it is however important to deal with various forms of inconsistencies and incoherence in ABoxes and TBoxes in an integrated way.
- *Complexity:* Reasoning with expressive Description Logics typically is already intractable for standard reasoning tasks. Often the approaches for dealing with inconsistencies introduce an additional level of complexity. In order to assure practicability, these complexity issues need to be taken into account.
- *Support for Multiple/networked Ontologies:* Many approaches to dealing with inconsistencies have been developed for dealing with single, isolated ontologies. Few approaches have been developed specifically for dealing with multiple ontologies that are networked or distributed in a certain way. We evaluate what kind of networking relationships are supported or how the approach can be extended to operate with multiple ontologies.

- *Exploitation of context or background knowledge:* Typical approaches for dealing with inconsistencies only consider the content of the knowledge base itself as input for diagnosis and repair. However, often it is useful to consider additional information about the relevance and importance of particular parts of the knowledge base as background knowledge. Such context information may be captured as provenance (e.g., indicating the trustworthiness of the source), in the form arguments (e.g., why certain axioms have been introduced), etc.
- *Interactivity, user involvement:* Many approaches to dealing with inconsistencies aim at a completely automated procedure. Others rely on the user to decide how to deal with particular situations. For example, it may be possible that the diagnosis of the problem is performed automatically, but the decision about how to fix a problem may be left to the user.
- *Availability of implementations:* Finally we discuss whether the approach is implemented and available for use or whether it would be feasible and desirable to implement it.

## 3.2 Overview of Approaches

In this section, we give an overview of the approaches for resolving inconsistency. When resolving inconsistency, we can either delete some erroneous axioms or weaken them. In any case, we often expect that minimal information is dropped to restore inconsistency. To achieve this requirement of minimal change, we often need a technique called debugging, which we will introduce in the following. There are mainly two important groups working on debugging and inconsistency (or incoherence) handling. The first group comes from Vrije Universiteit Amsterdam and the second group is the MidSwap group at University of Maryland. There are other work on resolving inconsistency which discuss specific scenario such as ontology revision and ontology integration. In the following, we first introduce the approaches for debugging and diagnosis. After that, we give a brief review of the application of AGM's belief revision theory to DLs. Finally, we introduce the approaches for knowledge integration in DLs.

### 3.2.1 Debugging and Diagnosis in DL-based Ontologies

We introduce two approaches given in [SHC06] which are proposed to debug an ontology, i.e. to calculate MUPS and MIPS: a top-down approach and an informed bottom-up approach.

**A top-down approach to explanation:** The first approach is originally proposed in [SC03]. Their debugging approach is restricted to unfoldable $\mathcal{ALC}$ TBoxes. Suppose $\mathcal{T}$ is an incoherent unfoldable TBox and $A$ is an unsatisfiable in it. To calculate a MUPS of $\mathcal{T}\ w.r.t\ A$, we can construct a tableau from a branch $B$ initially containing only *labelled formula* $(a\ :\ A)^{\emptyset}$ (for a new individual name $a$) by applying the tableau rules as long as possible. The rules are standard $\mathcal{ALC}$-tableau rules with lazy unfolding, and have to be read as follows: assume that there is a tableau $T = \{B, B1, ..., B_n\}$ with $n + 1$ branches. After applying one of the rules on $B$, we get a tableau $T' = \{B', B_1, ..., B_n\}$ or $T'' = \{B', B'', B_1, ..., B_n\}$.

Once no more rules can be applied, we know which atoms are needed to close a saturated branch and can construct a minimisation function for $A$ and $\mathcal{T}$ according to the tableau rules. A propositional formula $\phi$ is called a *minimisation function for $A$ and $\mathcal{T}$* if $A$ is unsatisfiable in every subset of $\mathcal{T}$ containing the axioms which are true in an assignment making $\phi$ true. Here axioms are used as propositional variable in $\phi$. As we can identify unsatisfiability of $A\ w.r.t$ a set $S$ of axioms with a closed tableau using only the axioms in $S$ for unfolding, branching on a disjunctive rule implies that we need to join the functions of the appropriate sub-branches conjunctively. If an existential rule has been applied, the new branch $B'$ might not necessarily be closed on formulas for both individuals. Assume that $B'$ closes on the individual $a$ but not on $b$. In this case $min\_function(a, B, \mathcal{T}) = \bot$, which means that the related disjunct does not influence the calculation of the minimal incoherent TBox.

Based on the minimisation function $min\_function(a, \{(a\ :\ A)^{\emptyset}\}, \mathcal{T})$, denoted $\phi$, which is calculated using some rules, we then can calculate the MUPS of $\mathcal{T}\ w.r.t\ A$.

From MUPS we can easily calculate MIPS based on an additional operation on sets of TBoxes, called *subset-reduction*. Let $M = \{\mathcal{T}_1, ..., \mathcal{T}_m\}$ be a set of TBoxes. The *subset-reduction* of $M$ is the smallest subset $sr(M) \subseteq M$ such that for all $\mathcal{T} \in M$ there is a set $\mathcal{T}' \in sr(M)$ such that $\mathcal{T}' \subseteq \mathcal{T}$.

Let $\mathcal{T}$ be an incoherent TBox with unsatisfiable concepts $\Delta^{\mathcal{T}}$. The set of all MIPSs of $\mathcal{T}$, denoted $mips(\mathcal{T})$, is obtained by the following equation: $mips(\mathcal{T}) = sr(\bigcup_{A \in \Delta^{\mathcal{T}}} mups(\mathcal{T}, A))$, where $mups(\mathcal{T}, A))$ is the set of all MUPSs of $\mathcal{T}$ $w.r.t$ $A$.

**A bottom-up approach to explanation:** The top-down approach is based on modifying the internals of a DL reasoner. This approach is computationally very hard in the worst-case. In [SHC06], a bottom-up approach is proposed to calculate MUPS with the support of an external DL reasoner. The main advantage of this approach is that it can deal with any DL-based ontology supported by an external reasoner. Unlike the top-down approach, they support various DL-based ontology languages, including OWL-DL.

Given an unsatisfiable concept $A$ and a terminology $\mathcal{T}$, MUPS can be systematically calculated by checking whether $A$ is unsatisfiable in subsets $\mathcal{T}'$ of $\mathcal{T}$ of increasing size. Such a procedure is complete and easy to implement, but infeasible in practice because the number of the subsets of $\mathcal{T}$ is exponential to the number of axioms in $\mathcal{T}$. To solve this problem, a *selection function* is introduced to control the subsets of $\mathcal{T}$ that are checked for satisfiability of $A$. Such a selection function selects increasingly large subsets which are heuristically chosen to be relevant additions to the currently selected subset. Although this approach is not guaranteed to give us the complete solution set of MUPS, it provides an efficient approach for debugging inconsistent terminologies.

**Calculating terminological diagnoses:** Terminological diagnosis, as defined in [Sch05], is an instance Reiter's diagnosis from first principles. Therefore, we can use Reiter's algorithms to calculate terminological diagnoses. An important notion in diagnosis is called a *conflict set*, which is an incoherent subset of a TBox. Given a TBox $\mathcal{T}$, a subset $\mathcal{T}'$ of $\mathcal{T}$ is a diagnosis for an incoherent $\mathcal{T}$ if $\mathcal{T}'$ is a minimal set such that $\mathcal{T} \setminus \mathcal{T}'$ is not a conflict set for $\mathcal{T}$.

Reiter introduced a hitting set tree algorithm to calculate diagnoses from conflict sets [Rei87]. Given a collection of sets $\mathcal{C}$, a *hitting set* for $\mathcal{C}$ is a set $H \subseteq \cup_{S \in \mathcal{C}} S$ such that $H \cap S \neq \emptyset$ for each $S \in \mathcal{C}$. A hitting set $H$ for $\mathcal{C}$ is minimal if and only if the following conditions hold: (1) $H$ is a hitting set; (2) for any $H' \in \mathcal{C}$, if $H \subset H'$, then $H'$ is a hitting set for $\mathcal{C}$. It has been shown in [SHC06] that a subset $\mathcal{T}'$ of $\mathcal{T}$ is a diagnosis for an incoherent TBox $\mathcal{T}$ if and only if $\mathcal{T}'$ is a minimal hitting set for the collection of conflict sets of $\mathcal{T}$.

To calculate minimal hitting sets, we can adapt Reiter's hitting set tree (HS-tree) algorithm. Given a collection $\mathcal{C}$ of sets, a HS-tree $T$ is the smallest edge-labeled and node-labeled tree, such that the root is labeled by ✓ if $\mathcal{C}$ is empty. Otherwise it is labeled with any set in $\mathcal{C}$. For each node $n$ in $T$, let $H(n)$ be the set of edge labels on the path in $T$ from the root to $n$. The label for $n$ is any set $S \in \mathcal{C}$ such that $S \sqcap H(n) = \emptyset$, if such a set exists. If $n$ is labeled by a set $S$, then for each $\sigma \in S$, $n$ has a successor, $n_\sigma$ joined to $n$ by an edge labeled by $\sigma$. For any node labeled by ✓, $H(n)$, i.e. the labels of its path from the root, is a hitting set for $\mathcal{C}$.

Figure 3.1 shows a HS-tree $T$ for the collection $\mathcal{C} = \{\{1,2,3,4,5,6\}, \{3,4,5\}, \{1,2,4,6\}, \{1,2\}, \{4,7\}\}$ of sets. $T$ is created breadth first, starting with root node $n_0$ labeled with $\{1,2,3,4,5,6\}$. For diagnostic problems the sets in the collection are conflict sets which are created on demand. In our case, conflict sets for a terminological diagnosis problem can be calculated by a standard DL engine (by definition each incoherent subset of $\mathcal{T}$ is a conflict set).

### 3.2.2 Debugging and Repairing OWL Ontologies in SWOOP

In [PSK05, KPGS06], two orthogonal debugging approaches are proposed to detect the clash/sets of support axioms responsible for an unsatisfiable classes, and to identify root/derived unsatisfiable classes. The first one is a glass box approach which is based on description logic tableaux reasoner-Pellet. This approach is closely related to the top-down approach to explanation in [SHC06]. However, the approach proposed in [PSK05] is not limited to DL $\mathcal{ALC}$ and is designed for OWL DL. The second one is a black box approach [KPGS06] which is better suitable to identify dependencies in a large number of unsatisfiable classes. The approach is reasoner-independent, in the sense that the DL reasoner is solely used as an oracle to determine

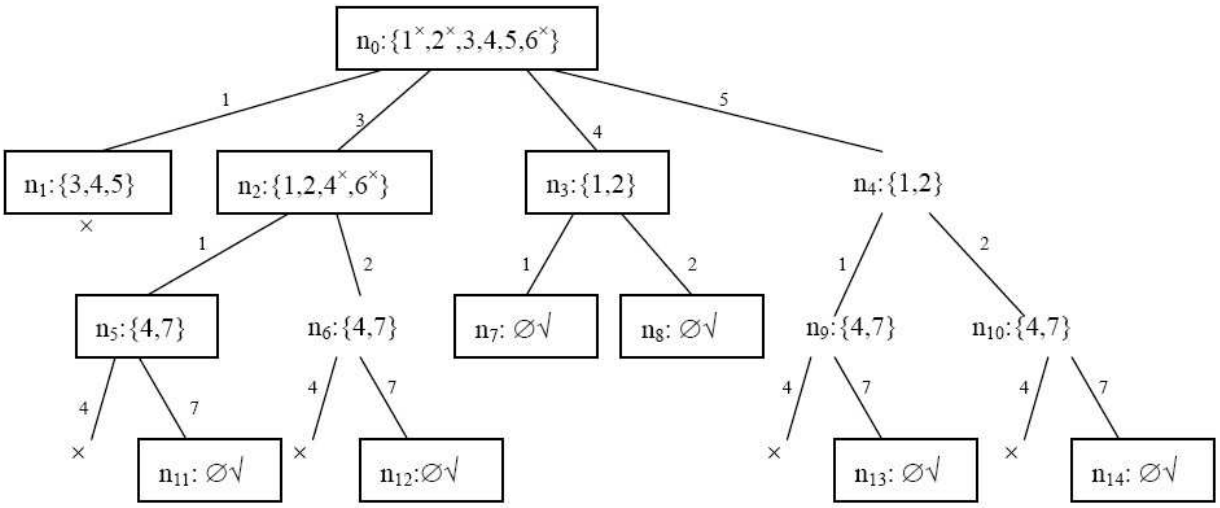


Figure 3.1: HS-Tree with small conflict sets

concept satisfiability with respect to a TBox. It consists of two main steps. In the first step, it computes a *single* MUPS of the concept and then it utilizes the Hitting Set algorithm to retrieve the remaining ones. This approach is closely related to the bottom up approach to explanation. Based on the debugging approach, in [KPSG06], the authors give a tool to repair unsatisfiable concepts in OWL ontologies. The basic idea is to rank erroneous axioms and then to generate a plan to resolve the errors in a given set of unsatisfiable concepts by taking into account the axiom ranks.

### 3.2.3 AGM and Description Logics

AGM's theory of belief change [Gar88] has been widely used to deal with logical inconsistency resulting from revising a knowledge base by newly received information. There are three types of belief change, i.e. *expansion*, *contraction* and *revision*. Expansion is simply to add a sentence to a knowledge base; contraction requires to consistently remove a sentence from a knowledge base and revision is the problem of accommodating a new sentence to a knowledge base consistently. Alchourrón, Gardenfors and Markinson proposed a set of postulates to characterize each belief change operator. The application of AGM' theory to description logics is not trivial because it is based on the assumptions that generally fail for DLs [FPA04]. For example, a DL is not necessarily closed under the usual operators such as $\neg$ and $\wedge$. In [FPA05, FPA06], the basic AGM postulates for contraction were generalized to DLs and the feasibility of applying the generalized AGM theory of contraction to DLs and OWL was studied. There work is based on the *coherence model*. That is, the knowledge base is closed under consequence operation, i.e., $K = Cn(K)$, where $K$ is a knowledge base and $Cn$ is the consequence operation of the underlying language. They showed that in many important DLs, such as $\mathcal{SHOIN}(\mathcal{D})$ and $\mathcal{SHIQ}$, it is impossible to define a contraction operator that satisfies the generalized AGM postulates. However, they didn't apply AGM's postulates for a revision operator and explicit construction of a revision operator was not considered in their paper.

The original AGM theory of belief revision is based on the coherence model. However, this causes problems in practice [Fuh91] and the foundational model was then proposed. Under this model, there is a clear distinction between information in the knowledge base and information which can be inferred from the knowledge base. In [FPA06] and [QLB06a], the problem of applying AGM theory of belief revision to DLs under foundational model is discussed. In [FPA06], the authors introduced the concept of negation to DLs and then generalized AGM postulates for contraction and revision. However, the authors do not consider the construction of a revision operator in their paper. In contrast, the work in [QLB06a] generalized the revised postulates for belief revision in [KM92] and proposed two revision operators which satisfy the generalized postulates.

One operator is the weakening-based revision operator which is defined by weakening of statements in a DL knowledge base. The weakening-based revision operator may result in counterintuitive results in some cases, so another operator was proposed to refine it. It was shown that both operators capture some notions of minimal change.

### 3.2.4 Knowledge integration for description logics

In [MLB05], an algorithm, called *refined conjunctive maxi-adjustment* (RCMA for short) was proposed to weaken conflicting information in a *stratified* DL knowledge base and some consistent DL knowledge bases were obtained. To weaken a terminological axiom, they introduced a DL expression, called *cardinality restrictions* on concepts. However, to weaken an assertional axiom, they simply delete it. In [QLB06b], the authors first define two revision operators in description logics, one is called a weakening-based revision operator and the other is its refinement. The revision operators are defined by introducing a DL constructor called *nominals*. The idea is that when a terminology axiom or a value restriction is in conflict, they simply add explicit exceptions to weaken it and assume that the number of exceptions is minimal. Based on the revision operators, they then propose an algorithm to handle inconsistency in a *stratified* description logic knowledge base. It was shown that when the weakening-based revision operator is chosen, the resulting knowledge base of their algorithm is semantically equivalent to that of the RCMA algorithm. However, their syntactical forms are different.

## 3.3 Discussion

We have introduced some existing approaches for resolving inconsistency and incoherence in DLs. In this section, we compare these approaches with respect to the evaluation criteria proposed in Section 3.1. The results of comparison are compactly summarized in Table 3.1. According to Table 3.1, MUPSter and SWOOP are mainly applied to debug and repair incoherence, whilst AGM-based approaches and knowledge integration approaches are applied to deal with inconsistency. When resolving incoherence, MUPSter will delete axioms in TBox and SWOOP deletes either axioms or concepts in TBox. To resolve inconsistency, the AGM-based approaches and knowledge integration approaches either delete axioms in a DL knowledge base or remove some instances which are responsible for inconsistency. Before dealing with incoherence, both MUPster and SWOOP may split the axioms into smaller axioms, so the structure will be lost. One of the AGM-based approach given in [QLB06a] also requires to split axioms, so it does not preserve the structure of the axioms. The knowledge integration approach proposed in [MLB05] needs to transform all terminology axioms into the cardinality restrictions on concepts. So the structure of the axiom is lost. It has been shown that debugging in DL $\mathcal{ALC}$ is PSPACE-complete in [SC03] because it is based on tableux algorithm. The glass box approach is also based on tableau algorithm, so it is at least PSPACE-hard. Other approaches are at least PSPACE-hard because they need to checking inconsistency, which is PSPACE-hard. Among all the approaches, only MUPSter and SWOOP are implemented and only SWOOP has user interface. The MUPSter and AGM-based approach do not explore context information to deal with incoherence or inconsistency. Whilst SWOOP and knowledge integration approaches explore the ranking information to resolve incoherence or inconsistency.

Table 3.1: Evaluation results

| Criteria | MUPSter | SWOOP | AGM-based approaches | knowledge integration in DLs |
|---|---|---|---|---|
| Application | debugging, repair | debuging, repair | revision | merging |
| Granularity | axiom | axiom or concept | axiom or instance | axiom or instance |
| Preservation of structure | partially | partially | partially | partially |
| Support for ABox, TBox | TBox | TBox | TBox and ABox | TBox and ABox |
| Inconsistency vs. Incoherence | incoherence | incoherence | inconsistency | inconsistency |
| Complexity | PSPACE-complete | PSPACE-hard | PSPACE-hard | PSPACE-hard |
| User involvement | no | yes | no | no |
| Availability of implementation | yes | yes | no | no |
| Support for networked ontologies | no | no | no | yes |
| Exploitation of context | no | partially | no | yes |

# Chapter 4

# Resolving Inconsistency and Incoherence

## 4.1 Inconsistency and Incoherence

The relationship between incoherence and inconsistency is not simple. Firstly, the fact that an ontology is inconsistent does not necessarily imply that it is incoherent, and vice versa. There exist different combinations of inconsistency and incoherence, as illustrated in Figure 4.1 and discussed in the following.

Figure 4.1 (1) is an example of a consistent but incoherent ontology, in which the two disjoint concepts $C1$ and $C2$ share a sub-concept $C3$. Figure 4.1 (2)-(4) show examples of inconsistent ontologies. Figure 4.1 (2) is an example of an inconsistent and incoherent ontology, in which the two disjoint concepts $C1$ and $C2$ share a sub-concept which is a nominal $\{a\}$. Figure 4.1 (3) is an example of an inconsistent ontology in which an instance $a$ instantiates a concept $C1$ and its complement $\neg C1$. Figure4.1 (4) is an example of an inconsistent but coherent ontology, in which the two disjoint concepts $C1$ and $C2$ share an instance $a$. Finally, Figure 4.1 (5) shows an example of an ontology that is both incoherent and inconsistent.

From the definitions of incoherence above, we know that incoherence can occur in the terminology level only. When dealing with inconsistency, we can differentiate terminology axioms and assertional axioms. We have the following categorization of different kinds of reason for inconsistent ontologies.

- *Inconsistency due to terminology axioms*: In this case, we only consider inconsistency in TBoxes. Figure 4.1 (2) is an example of such an inconsistency. Following our definitions, this kind of inconsistency will make the TBox incoherent.
- *Inconsistency due to assertional axioms:* This kind of inconsistency only occurs in ABoxes. It is not related to incoherence. A source of assertional inconsistency is that there are conflicting assertions about one individual, e.g., an individual is asserted to belongs to a class and its complement class, as in Figure 4.1 (2).
- *Inconsistency due to terminology and assertional axioms:* In this case, each conflicting set of axioms must contain both terminology axioms and assertional axioms. This kind of inconsistency is sometimes dependent on incoherence. Such an example is shown in Figure 4.1 (5). It is easy to see that $C_3$ is an unsatisfiable concept and that $O$ is inconsistent. The reason for the inconsistency is that the individual $a$ instantiates $C_3$ which is unsatisfiable. Therefore, if we repair the unsatisfiable concept $C_3$, then the inconsistency will disappear as well. On the other hand, the inconsistency in example in Figure 4.1 (4) is not caused by an incoherence.

The first kind of inconsistency is only related to terminology axioms. In this case, the unit of change is a concept (either atomic or complex). Therefore, some revision approaches which take the individual names as the unit of change, such as the one proposed in [QLB06a], cannot be applied to deal with this kind of inconsistency. By contrast, the other two kinds of inconsistency are related to assertional axioms. So the unit of change can be either a concept or an individual name.

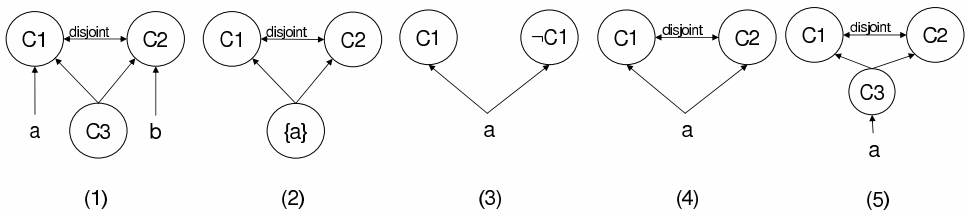


Figure 4.1: Examples of variants of inconsistency and incoherence.

From this discussion we observe that the causes for incoherence and inconsistency are manifold and their interdependencies are complex. Incoherence is always caused by conflicts in the terminology. It may or may not affect the consistency of the overall ontology. Inconsistency may arise due to conflicts in the ABox, in the TBox, or a combination of both ABox and TBox. In the following section we propose a revision approach resolving conflicts in evolving ontologies that takes these interdependencies into account.

## 4.2 Resolving Inconsistency and Incoherence

The problem of ontology revision is described as follows. Suppose we have two ontologies $O = \langle \mathcal{T}, \mathcal{A} \rangle$ and $O' = \langle \mathcal{T}', \mathcal{A}' \rangle$ where $O$ is the original ontology and $O'$ is the newly received ontology which contains a set of axioms to be added to $O$. Even if both $O$ and $O'$ are individually consistent and coherent, putting them together may cause inconsistency or incoherence. Therefore, we need to delete or weaken some axioms in $O$ to restore consistency and coherence. Usually, the result of revision is a set of ontologies rather than a unique ontology [QLB06a]. More formally, we have the following definition of ontology revision. We denote all the possible ontologies with $\mathcal{O}$.

We first introduce the notion of a disjunctive ontology from [MLB05]. A disjunctive ontology, denoted as **O**, is a set of ontologies. The semantics of the disjunctive ontology is defined as follows [MLB05]:

**Definition 7** *A disjunctive ontology* $\mathbf{O}$ *is satisfied by an interpretation* $\mathcal{I}$ *(or* $\mathcal{I}$ *is a model of* $\mathbf{O}$*) iff* $\exists O \in \mathbf{O}$ *such that* $\mathcal{I} \models O$*.* $\mathbf{O}$ *entails* $\phi$*, denoted* $\mathbf{O} \models \phi$*, iff every model of* $\mathbf{O}$ *is a model of* $\phi$*.*

**Definition 8** *An ontology revision operator (or revision operator for short) in DLs is a function* $\circ : \mathcal{O} \times \mathcal{O} \rightarrow \mathcal{P}(\mathcal{O})$ *which satisfies the following conditions: 1) suppose that* $\mathcal{P}(\mathcal{O})$ *denotes all the subsets of* $\mathcal{O}$*,* $O \circ O' \models \phi$ *for all* $\phi \in O'$*; 2) for each* $O_i \in O \circ O'$*,* $O_i$ *is consistent.*

That is, an ontology revision operator is a function which maps a pair of ontologies to a disjunctive ontology which can consistently infer the newly received ontology. In practice, we may only need one ontology after revision. In this case, we can obtain such an ontology by ranking the ontologies obtained by the revision operator and then selecting the one with highest rank. Ranking of ontologies can either be given by the users or be computed by some algorithms.

The current work on ontology revision suffers from some problems, to name a few, we have the following ones:

- There is much work on the analysis of applicability of AGM postulates for belief change to DLs [FPA05, FHP$^+$06]. However, few of them discuss the concrete construction of a revision approach.
- Current revision approaches often focus on dealing with logical inconsistency. Another problem which is as important as inconsistency handling is incoherence handling, where an ontology is incoherent if and only if there is an unsatisfiable named concept in its terminology. As analyzed in [FHP$^+$06], logical incoherence and logical inconsistency are not independent of each other. A revision approach which resolves both logical incoherence and inconsistency is missing.

We now propose our general approach which resolves incoherence and inconsistency in an integrated way. The approach consists of the process steps shown in Figure 4.2. In this process, problems that are related only with either the TBox or the ABox are dealt with independently in two separate threads (c.f. left and right thread of Figure 4.2, respectively). For the TBox, inconsistency resolution is done before incoherence resolution because incoherence is a consequence of inconsistency in the TBox. We first check if $\mathcal{T} \cup \mathcal{T}'$ is consistent. If it is not, then we resolve inconsistency. This can be done by either deleting the erroneous terminology axioms or weakening them. In a next step, we resolve incoherence. There are several ways to resolve incoherence. The commonly used technique is to remove some (usually minimal numbers) of erroneous terminology axioms which are responsible for the incoherence. Alternatively, we can take the

maximal coherent sub-ontologies of $\mathcal{T}$ w.r.t $\mathcal{T}'$ as the result of revision [MLBP06, LPSV06]. For the ABox, we resolve inconsistencies that occur only due to assertional axioms. This can be done by either deleting the assertional axioms which are responsible for the inconsistency or weakening them. The weakening of assertional axioms may be different from that of terminology axioms. Finally, we deal with the inconsistency due to both terminology and assertional axioms. In each of the revision steps, the result may be a disjunctive ontology, since there may exist several alternative way to resolve the incoherence or inconsistency. However, in each step a decision is made, which single ontology should be selected as input for the subsequent step. This decision can be made either by the user or an automated procedure based on a ranking of the results as discussed above.

Our general approach does not yet specify *how* to deal with inconsistency or incoherence. Moreover, for different kinds of inconsistency, we can use different strategies to resolve them. For example, when resolving inconsistency due to terminology axioms, we can take the maximal consistent subsets of the original TBox $w.r.t$ the new TBox as the result of revision. Whilst when resolving inconsistency related to assertional axioms, we can apply the revision approach in [QLB06a], which removes minimal number of individual resulting in the conflict. In the next section, we instantiate our approach by proposing a concrete approach to resolving incoherence and some concrete approaches to resolving inconsistency.

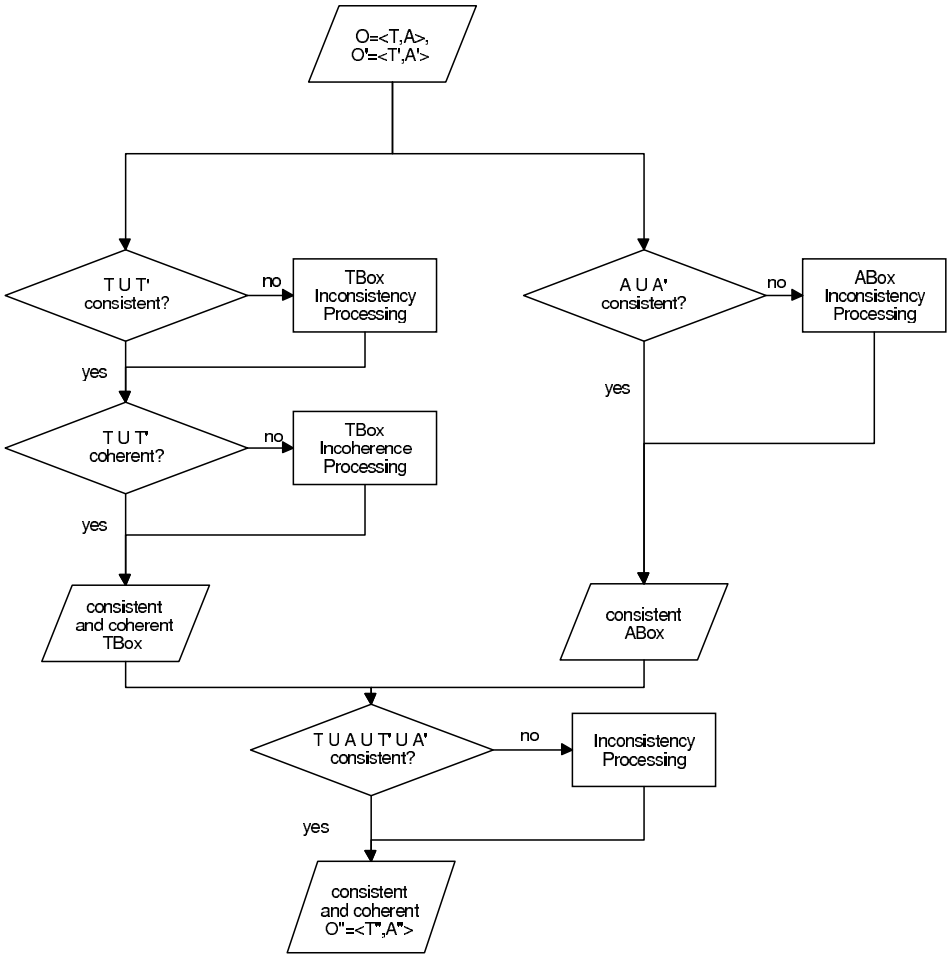


Figure 4.2: Approach to resolving inconsistency and incoherence

# Chapter 5

# Instantiation of the General Approach

When resolving incoherence or inconsistency, we need to delete or weaken some axioms in the original ontology. In this case, we want to keep as much information in the original ontology as possible. There are several ways to achieve this. For example, to resolve incoherence, we may require to delete a minimal number of axioms to get a coherent ontology or we take a maximal consistent sub-ontology as the result of revision. In the following, we propose some concrete approaches to resolving incoherence and resolving inconsistency which capture some kinds of minimal change. When resolving incoherence or inconsistency in a TBox, the unit of change can be either an axiom or a concept. However, if there are axioms in ABox which are responsible for inconsistency, the unit of change can be either an individual, a concept, or an axiom. In the following, we propose an approach for resolving incoherence by removing a minimal number of terminology axioms and two approaches for resolving inconsistency by weakening a concept.

## 5.1 Resolving incoherence

In this section, we propose an approach to resolving incoherence in ontology evolution. Resolving incoherence is a problem which is often overlooked during ontology evolution. This problem is easy to be confused with resolving inconsistency. Many debugging approaches have been proposed to pinpoint the unsatisfiable concepts or terminology axioms which are responsible for incoherence. There are mainly two ways to resolve the logical incoherence. The first way is to remove some (usually minimal numbers) of erroneous terminology axioms which are responsible for the incoherence to restore coherence. Alternatively, we can take the maximal coherent sub-ontologies of $O\ w.r.t\ O'$ as the result of revision [MLBP06]. Our approach to resolve incoherence belongs to the first class, i.e. we delete some terminology axioms to restore coherence.

We first generalize the concept of *minimal incoherence-preserving sub-TBox* (MIPS) defined in [SC03].

**Definition 9** *Let $\mathcal{T}$ and $\mathcal{T}_0$ be two TBoxes, where $\mathcal{T}$ is an old TBox and $\mathcal{T}_0$ is a newly received TBox. A minimal incoherence-preserving sub-TBoxes (MIPS) $\mathcal{T}'$ of $\mathcal{T}\ w.r.t\ \mathcal{T}_0$ is a sub-Tbox of $\mathcal{T}$ which satisfies (1) $\mathcal{T}' \cup \mathcal{T}_0$ is incoherent; (2) $\forall \mathcal{T}'' \subset \mathcal{T}'$, $\mathcal{T}'' \cup \mathcal{T}_0$ is coherent. We denote the set of all MIPS of $\mathcal{T}\ w.r.t\ \mathcal{T}_0$ by $MIPS_{\mathcal{T}_0}(\mathcal{T})$.*

A MIPS of a TBox $\mathcal{T}\ w.r.t$ TBox $\mathcal{T}_0$ is the minimal sub-TBox of $\mathcal{T}$ that is incoherent with $\mathcal{T}_0$. It is similar to the *kernel* defined in [Han94]. In classical logic, given a knowledge base $A$ which is a set of classical formulas and a formula $\phi$, a $\phi$-kernel of $A$ is the minimal subbase of $A$ that implies $\phi$ [Han94]. To define a contraction function called kernel contraction, Hansson defines an incision function which selects formulas to be discarded in each $\phi$-kernel of $A$. We adapt the incision function to define our revision operator.

**Definition 10** *Let $\mathcal{T}$ be a TBox. An* incision function *for $\mathcal{T}$, denoted as $\sigma$, is a function such that for each TBox $\mathcal{T}_0$*

*(i)* $\sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) \subseteq \bigcup_{\mathcal{T}_i \in MIPS_{\mathcal{T}_0}(\mathcal{T})} \mathcal{T}_i$;

*(ii) if* $\mathcal{T}' \in MIPS_{\mathcal{T}_0}(\mathcal{T})$ *and* $\mathcal{T}' \neq \emptyset$*, then* $\mathcal{T}' \cap \sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) \neq \emptyset$.

That is, an incision function for a TBox $\mathcal{T}$ is a function such that for each TBox $\mathcal{T}_0$, it selects formulas from every MIPS of $\mathcal{T}$ $w.r.t$ $\mathcal{T}_0$ if this MIPS is not empty. The incision function plays a similar role as concept pinpointing in [SC03].
An important incision function is the one which is called *minimal incision function* [FFKI06]. We redefine it as follows.

**Definition 11** *Let* $\mathcal{T}$ *be a TBox. An incision function* $\sigma$ *for* $\mathcal{T}$ *is* minimal *if there is no other incision function* $\sigma'$ *such that there is a TBox* $\mathcal{T}_0$, $\sigma'(MIPS_{\mathcal{T}_0}(\mathcal{T})) \subset \sigma(MIPS_{\mathcal{T}_0}(\mathcal{T}))$.

We give a definition which refines the minimal incision function.

**Definition 12** *Let* $\mathcal{T}$ *be a TBox. An incision function* $\sigma$ *for* $\mathcal{T}$ *is* cardinality-minimal *if there is no other incision function* $\sigma'$ *such that there is a TBox* $\mathcal{T}_0$, $|\sigma'(MIPS_{\mathcal{T}_0}(\mathcal{T}))| < |\sigma(MIPS_{\mathcal{T}_0}(\mathcal{T}))|$.

A cardinality-minimal incision function is always a minimal incision function.

**Proposition 1** *Let* $\mathcal{T}$ *be a TBox. Suppose* $\sigma$ *is a cardinality-minimal incision function for* $\mathcal{T}$*, then it is a minimal incision function.*

The proof of Proposition 1 is clear by considering Definition 11 and Definition 12.
From each incision function, we can define a revision operator.

**Definition 13** *Let* $\mathcal{T}$ *be a TBox, and* $\sigma$ *be an incision function for* $\mathcal{T}$*. The* kernel revision *operator* $\circ_\sigma$ *for* $\mathcal{T}$ *is defined as follows: for each TBox* $\mathcal{T}_0$*,*

$$\mathcal{T} \circ_\sigma \mathcal{T}_0 = \{(\mathcal{T} \setminus \sigma(MIPS_{\mathcal{T}_0}(\mathcal{T}))) \cup \mathcal{T}_0\}.$$

The resulting TBox of the kernal revision operator only contains one TBox. For simplicity, we write $\mathcal{T} \circ_\sigma \mathcal{T}_0 = (\mathcal{T} \setminus \sigma(MIPS_{\mathcal{T}_0}(\mathcal{T}))) \cup \mathcal{T}_0$ later. According to the definition of an incision function, the resulting TBox of the kernel revision operator is always a coherent TBox.
The kernel revision operator is defined via an incision function. There are many ways to give an incision function. For example, we can apply Reiter's hitting set algorithm [Rei87] to compute the cardinality-minimal incision function.

**Example 1** *We consider a small TBox* $\mathcal{T} = \{Employee \sqsubseteq Person, Student \sqsubseteq Person, PhDStudent \sqsubseteq Student, Student \sqsubseteq \neg Employee, Article \sqsubseteq Publication, Article \sqsubseteq \forall author.Person\}$*. Now consider a TBox* $\mathcal{T}_0 = \{PhDStudents \sqsubseteq Employee\}$ *that is to be added to the* $\mathcal{T}$*. The union* $\mathcal{T} \cup \mathcal{T}_0$ *is incoherent. The TBox* $\mathcal{T}' = \{PhDStudent \sqsubseteq Student, Student \sqsubseteq \neg Employee\}$ *is a (the only) minimal incoherence sub-TBox of* $\mathcal{T}$ *w.r.t. to* $\mathcal{T}_0$*. As there exists only one MIPS w.r.t.* $\mathcal{T}_0$*, a cardinality-minimal incision function would select either one of the axioms in* $\mathcal{T}'$ *to be removed for resolving the incoherence.*

## 5.2 Resolving inconsistency

### 5.2.1 Inconsistency due to terminology axioms

Many approaches for resolving inconsistency in classical logic are based on some maximal consistent subsets, e.g. the cardinality-maximizing revision approach in [Gin86]. The idea is that we select all the cardinality-maximizing subsets of the original knowledge base that are consistent with the new knowledge base. In this subsection, we apply the cardinality-maximizing revision approaches to dealing with terminological inconsistency.

**Definition 14** *Let $\mathcal{T}$ and $\mathcal{T}'$ be two TBoxes. A subset $\mathcal{T}_i$ of $\mathcal{T}$ is said to be cardinality-maximizing consistent $w.r.t\ \mathcal{T}'$ if and only if it satisfies the following conditions:*

*(m1) $\mathcal{T}_i \cup \mathcal{T}' \not\models \bot$ and*

*(m2) $\forall \mathcal{T}_j \subseteq \mathcal{T}$, if $|\mathcal{T}_i| < |\mathcal{T}_j|$ then $\mathcal{T}_j \cup \mathcal{T}' \models \bot$, where $|\mathcal{T}_i|$ is the cardinality of the set $\mathcal{T}_i$.*

*The set of all cardinality-maximal consistent sub-ontologies of $\mathcal{T}\ w.r.t\ \mathcal{T}'$ is denoted by $CMAXCON(\mathcal{T})_{\mathcal{T}'}$.*

Condition m1 says that $\mathcal{T}_i$ is consistent with $\mathcal{T}'$ and Condition m2 states that any sub-ontology of $\mathcal{T}$ that contains more elements than $\mathcal{T}_i$ is in inconsistent with $\mathcal{T}'$.
We define a revision operator based on cardinality-maximizing consistent sub-TBoxes.

**Definition 15** *Let $\mathcal{T}$ and $\mathcal{T}'$ be two TBoxes. Then the cardinality-maximizing consistent sub-TBoxes based revision operator, denoted as $\circ_{CM}$, is defined as follows:*

$$\mathcal{T} \circ_{CM} \mathcal{T}' = \{\mathcal{T}_i \cup \mathcal{T}' : \mathcal{T}_i \in CMAXCON(\mathcal{T})_{\mathcal{T}'}\}.$$

**Example 2** *We consider a small TBox $\mathcal{T} = \{Employee \sqsubseteq Person, Student \sqsubseteq Person, PhDStudent \sqsubseteq Student, Student \sqsubseteq \neg Employee, \{paul\} \sqsubseteq PhDStudent$. Now consider a new TBox $\mathcal{T}' = \{\{paul\} \sqsubseteq Employee\}$ that is to be added to $\mathcal{T}$. The resulting terminology $\mathcal{T} \cup \mathcal{T}'$ is coherent, but inconsistent. There exist a number of cardinality-maximizing consistent sub-TBoxes of $\mathcal{T}$ w.r.t. $\mathcal{T}_0$ that can be obtained be removing either one of the following axioms from $\mathcal{T}$: $PhDStudent \sqsubseteq Student, Student \sqsubseteq \neg Employee, \{paul\} \sqsubseteq PhDStudent$.*

### 5.2.2 Inconsistency due to assertional axioms

There are two ways to resolve inconsistency due to assertional axioms: removing some axioms or weakening them. The cardinality-maximizing consistent sub-ontologies based revision approaches belongs to the first category. In the following, we consider how to weaken an assertional axiom.

**Definition 16** *Let $\phi$ be an assertional axiom. A weakened assertion $\phi_{weak}$ of $\phi$ is an assertion which satisfies $\phi \models \phi_{weak}$.*

More specifically, we have the following definitions for a weakened assertional axiom.

**Definition 17** *Let $O = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology and $\phi = R(a, b)$ be a role assertion in $\mathcal{A}$. A weakened relation assertion $\phi_{weak}$ of $\phi$ can have the following forms:*
*(R1) $\phi_{weak} = \emptyset$, where $\phi_{weak} = \emptyset$ means $\phi$ is deleted, or*
*(R2) $\phi_{weak} = S(a, b)$, where $\mathcal{T} \models R \sqsubseteq S$.*
*Let $\phi = a \approx b$ (or $a \not\approx b$) be an equality axiom (or an inequality axiom). Then $\phi_{weak} = \emptyset$.*

That is, to weaken a role assertion, we can either delete it or replace $R$ with a super-role. The weakening approaches are borrowed from [PT06].
Next, we give a new approach to weakening a concept assertion.

**Definition 18** *Let $O = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology and $\phi = C(a)$ be a concept assertion in $\mathcal{A}$. Then $\phi_{weak} = C_{weak}(a)$, where $C_{weak}$ is recursively defined as follows:*
*1) if $C = A$ or $\neg A$ for a concept name $A$, then*
*(C1) $C_{weak} = \top$, or*
*(C2) $C_{weak} = B$, where $\mathcal{T} \models C \sqsubseteq B$ and $B \neq \top$;*

*2) if* $C = C_1 \sqcap C_2$, *then* $C_{weak} = (C_1)_{weak} \sqcap (C_2)_{weak}$;
*3) if* $C = C_1 \sqcup C_2$, *then* $\phi_{weak} = (C_1)_{weak} \sqcup (C_2)_{weak}$;
*4) if* $C = \exists R.D$, *then*
*(C3)* $C_{weak} = \top$, *or*
*(C4)* $C_{weak} = \exists R.D \sqcup \{b_1, ..., b_n\}$, *where* $b_i \not\approx b_j$ *for* $i \neq j$;
*5) if* $C = \forall R.D$, *then*
*(C5)* $C_{weak} = \top$, *or*
*(C6)* $C_{weak} = \forall R.D \sqcup \{b_1, ..., b_n\}$, *where* $b_i \not\approx b_j$ *for* $i \neq j$;
*6) if* $C = \{b\}$, *where* $b$ *is an individual name, then*
*(C7)* $C_{weak} = \top$;
*7) if* $C = \geq nS$, *where* $S$ *is a simple role, then*
*(C8)* $C_{weak} = \top$, *or*
*(C9)* $C_{weak} = \geq mS$, *where* $m < n$;
*8) if* $C = \leq nS$, *where* $S$ *is a simple role, then*
*(C10)* $C_{weak} = \top$, *or*
*(C11)* $C_{weak} = \leq mS$, *where* $m > n$;

In Definition 18, to weaken an concept assertional axiom $C(a)$, we can simply weaken the concept $C$. If $C$ is a concept name, we either replace it by the top concept $\top$ or one of its upper concepts. If $C$ is the conjunction (or disjunction) of two concepts $C_1$ and $C_2$, then the weakening of $C$ is the conjunction (or disjunction) of the weakening of $C_1$ and that of $C_2$. If $C$ is of the form $\exists R.D$ (or $\forall R.D$), then we can weaken $D$ by adding some individuals to it. If $C$ is a nominal, then we simply replace it by $\top$. If $C$ is of the form $\geq nU$ (or $\leq nU$), where $U$ is where $U$ is either an abstract simple role or a concrete role, then we either replace it by $\top$ or lower (or raise) $n$.

The cardinality-maximizing consistent sub-TBoxes based revision operator capture the notion of minimal change by deleting minimal number of axioms. To define our weakening-based revision operator, we need to define the degree of a weakened assertion which can be used to measure the information loss of the revised ABox.

**Definition 19** *Let* $O = \langle \mathcal{T}, \mathcal{A} \rangle$ *be an ontology and* $\phi = R(a, b)$ *be a role assertion in* $\mathcal{A}$. *The degree of a weakened relation assertion* $\phi_{weak}$ *of* $\phi$ *can be defined as follows:*
*(1) if* $\phi_{weak} = \emptyset$, *then* $d(\phi_{weak}) = 2$;
*(2) if* $\phi_{weak} = S(a, b)$, *where* $\mathcal{T} \models R \sqsubseteq S$, *then* $d(\phi_{weak}) = 1$.
*Let* $\phi = a \approx b$ *(or* $a \not\approx b$*) be an equality axiom (or an inequality axiom). Then* $d(\phi_{weak}) = 1$.

In Definition 19, if a role assertion is deleted, then the degree of weakening is 2. However, if the role is replaced by a super-role, then the degree of weakening is 1. Therefore, suppose we want to weaken a role assertion, if we can find a super-role for $R$, then we replace $R$ by it because deleting a role assertion results in higher degree of weakening. Note that if $\mathcal{T}$ is empty, then we can only delete a role assertion if we want to weaken it.

**Definition 20** *Let* $O = \langle \mathcal{T}, \mathcal{A} \rangle$ *be an ontology and* $\phi = C(a)$ *be a concept assertion in* $\mathcal{A}$. *The degree of weakening of* $C_{weak}$, *denoted as* $d(C_{weak})$, *is recursively defined as follows:*
*1) suppose* $C = A$ *or* $\neg A$ *for a concept name* $A$,
*if* $C_{weak} = \top$ *then* $d(C_{weak}) = 2$,
*if* $C_{weak} = B$, *where* $\mathcal{T} \models C \sqsubseteq B$ *and* $B \neq \top$, *then* $d(C_{weak}) = 1$,
*2) if* $C = C_1 \sqcap C_2$, *then* $d(C_{weak}) = d((C_1)_{weak}) + d((C_2)_{weak})$;
*3) if* $C = C_1 \sqcup C_2$, *then* $d(C_{weak}) = max(d(C_1)_{weak}, d(C_2)_{weak})$;

*4) suppose* $C = \exists R.D$
*if* $C_{weak} = \top$ *then* $d(C_{weak}) = l$*, where* $l$ *is the number of individuals in* $\mathcal{A}$*,*
*if* $C_{weak} = \exists R.D \sqcup \{b_1, ..., b_n\}$ *then* $d(C_{weak}) = n$*;*
*5) suppose* $C = \forall R.D$*, then*
*if* $C_{weak} = \top$ *then* $d(C_{weak}) = l$*, where* $l$ *is the number of individuals in* $\mathcal{A}$*,*
*if* $C_{weak} = \forall R.D \sqcup \{b_1, ..., b_n\}$ *then* $d(C_{weak}) = n$*;*
*6) suppose* $C = \{b\}$*, where* $b$ *is an individual name, then* $d(C_{weak}) = 1$*;*
*7) suppose* $C = \geq nS$*, where* $S$ *is a simple role,*
*if* $C_{weak} = \top$ *then* $d(C_{weak}) = n$*,*
*if* $C_{weak} = \geq mU$*, where* $m < n$*, then* $d(C_{weak}) = n - m$*;*
*8) suppose* $C = \leq nS$*, where* $S$ *a simple role,*
*if* $C_{weak} = \top$ *then* $d(C_{weak}) = l$*, where* $l$ *is the number of individuals in* $\mathcal{A}$*,*
*if* $C_{weak} = \leq mS$*, where* $m > n$*, then* $d(C_{weak}) = m - n$*.*
*The degree of weakening of* $\phi$ *is* $d(\phi_{weak}) = d(C_{weak})$*.*

In Definition 20, suppose $C$ is a concept name or the negation of a concept name. If $C_{weak} = \top$, then the degree of weakening is 2. If $C_{weak}$ is the super-concept of $C$, then the degree of weakening is 1. Therefore, if we cannot find a super-concept (which is not $\top$) of $C$, we then replace $C$ by its super-concept and do not weaken it to $\top$. The degree of a conjunction is the sum of the degrees of its conjunct. A concept which is a disjunction, we use $max$ (instead of sum) to determine its degree of weakening. This definition agrees with the semantic interpretations of disjunction in many logics such as fuzzy logic and possibilistic logic. Suppose $C$ has the form $\exists R.D$ (or $\forall R.D$). If $C_{weak} = \top$, then we define that $d(\phi_{weak})$ is the number of individuals in $\mathcal{A}$. If $C_{weak} = \exists R.D \sqcup \{b_1, ..., b_n\}$ (or $C_{weak} = \forall R.D \sqcup \{b_1, ..., b_n\}$) then the degree of weakening of $\phi$ is $n$. If $C$ is a nominal then the degree of weakening is 1. Suppose $C$ has the form $\geq nS$. If $C_{weak} = \top$, then it is equivalent to say that the number $n$ is lowered to 0, so the degree of weakening is $n$. If $C_{weak} =\geq mS$, where $m < n$, then the degree of weakening is $n - m$. Suppose $C$ has the form $\leq nS$. If $C_{weak} = \top$, then it is equivalent to say that $n$ is raised to the number of individuals in $\mathcal{A}$, so the degree of weakening is $l$. If $C_{weak} =\leq mS$ where $m > n$, then the degree of weakening is $m - n$.
We define the degree of weakening of an ABox.

**Definition 21** *Let* $\mathcal{A}$ *be an ABox. Suppose* $\mathcal{A}_{weak}$ *is the ABox obtained by weakening axioms in* $\mathcal{A}$*. Then the degree of weakening of* $\mathcal{A}'$*, denoted as* $d(\mathcal{A}_{weak})$*, is defined as*

$$d(\mathcal{A}_{weak}) = \Sigma_{\phi_{weak} \in \mathcal{A}' \setminus \mathcal{A}} d(\phi_{weak}).$$

The degree of a weakened ABox is the sum of the degrees of all the weakened axioms.
A revision operator can be defined by weakening the axioms in the original ABox $\mathcal{A}$ $w.r.t$ the newly received ABox $\mathcal{A}'$. Let $Weak_{\mathcal{A}'}(\mathcal{A}) = \{\mathcal{A}_{weak} : \mathcal{A}_{weak} \cup \mathcal{A}' \; is \; consistent\}$.

**Definition 22** *Let* $O = \langle \mathcal{T}, \mathcal{A} \rangle$ *be an ontology and* $O' = \langle \mathcal{T}', \mathcal{A}' \rangle$ *is a newly received ontology. The weakening-based revision operator, denoted as* $\circ_{weak}$*, is defined as follows:*

$$\begin{aligned} \mathcal{A} \circ_{weak} \mathcal{A}' \;\; = \;\; & \{\mathcal{A}_{weak} \cup \mathcal{A}' : \mathcal{A}_{weak} \in Weak_{\mathcal{A}'}(\mathcal{A}) \; and \; \nexists \mathcal{A}_i \in Weak_{\mathcal{A}'}(\mathcal{A}), \\ & d(\mathcal{A}_i) < d(\mathcal{A}_{weak})\}. \end{aligned}$$

The result of revision of $\mathcal{A}$ $w.r.t$ $\mathcal{A}'$ is the set of ABoxes which are the unions of $\mathcal{A}'$ and the weakened ABoxes of $\mathcal{A}$ that are consistent with $\mathcal{A}'$ and have the minimal degree of weakening. The weakening-based revision operator cannot be applied to deal with inconsistency due to terminology axioms there is no individual in the TBox.

**Example 3** *Let* $\mathcal{T} = \{\forall teachesCourse.GraduateCourse(bob), teachesCourse(bob, cs401), GraduateCourse(cs401), teachesCourse(bob, cs402)\}$. *Suppose the newly received ontology is* $\mathcal{A}' = \{teachesCourse(bob, cs101), \neg GraduateClass(cs101)\}$. *It is clear that* $\mathcal{A} \cup \mathcal{A}'$ *is inconsistent. Since* $\forall teachesCourse.GraduateCourse(Bob)$ *is the only assertion axiom in* $\mathcal{A}$ *involved in conflict with* $\mathcal{A}'$*, we only need to weaken it to restore consistency, that is,*

$$\begin{aligned}\mathcal{A} \circ_w \mathcal{A}' = \{&\forall teachesCourse.(GraduateCourse \sqcup \{cs101\})(Bob),\\ &teachesCourse(bob, cs401), GraduateCourse(cs401), teachesCourse\\ &(bob, cs402), teachesCourse(bob, cs101), \neg GraduateClass(cs101)\}.\end{aligned}$$

### 5.2.3 Inconsistency due to terminology and assertional axioms

The inconsistency due to both terminology and assertional axioms is often caused by an unsatisfiable concept in a TBox and an assertion in an ABox that an individual belongs to this concept. The inconsistency due to an unsatisfiable concept and an assertion is related to incoherence and can be resolved after we deal with incoherence in a TBox. There are other reasons for the inconsistency due to terminology and assertional axioms. For example, suppose we have an ontology $O = \{bird \sqsubseteq flies, bird(Tweety)\}$ and another ontology $O' = \{\neg flies(Tweety)\}$. It is clear that $O \cup O'$ is inconsistent. To deal with this kind of inconsistency, we can either weaken the assertional axiom or terminology axioms (or both). We generalize the weakening-based approach to dealing with TBox.

**Definition 23** *Let* $O = \langle \mathcal{T}, \mathcal{A} \rangle$ *be an ontology. Suppose* $C \sqsubseteq D \in \mathcal{T}$*. A weakened GCI* $(C \sqsubseteq D)_{weak}$ *of* $C \sqsubseteq D$ *is defined as* $(C \sqsubseteq D)_{weak} = \top \sqsubseteq (\neg C \sqcup D)_{weak}$*, where* $(\neg C \sqcup D)_{weak}$ *is defined by Definition 18.*

In Definition 23, when we weaken a GCI $C \sqsubseteq D$, we first transform it to $\top \sqsubseteq \neg C \sqcup D$. Then we weaken the concept on the right side. The degree of weakening of $(C \sqsubseteq D)_{weak}$, denoted as $d((C \sqsubseteq D)_{weak})$, is equal to $d((\neg C \sqcup D)_{weak})$ (see Definition 20). The degree of weakening of a TBox is defined as follows.

**Definition 24** *Let* $O = \langle \mathcal{T}, \mathcal{A} \rangle$ *be an ontology. Then the degree of weakening of* $\mathcal{T}$*, denoted as* $d(\mathcal{T}_{weak})$*, is defined as*

$$d(\mathcal{T}_{weak}) = \Sigma_{(C \sqsubseteq D)_{weak} \in \mathcal{T}_{weak} \setminus \mathcal{T}} d((C \sqsubseteq D)_{weak}).$$

We generalize the weakening-based revision operator. Let $O_{weak} = \mathcal{T}_{weak} \cup \mathcal{A}_{weak}$, $d(O_{weak}) = d(\mathcal{T}_{weak}) + d(\mathcal{A}_{weak})$, and $Weak_{O'}(O) = \{O_{weak} : O_{weak} \cup \mathcal{T}' \cup \mathcal{A}' \ is \ consistent\}$.

**Definition 25** *Let* $O = \langle \mathcal{T}, \mathcal{A} \rangle$ *be an ontology and* $O' = \langle \mathcal{T}', \mathcal{A}' \rangle$ *be a newly received ontology. Suppose* $\mathcal{T} \cup \mathcal{T}$ *is consistent and coherent, and* $\mathcal{A} \cup \mathcal{A}'$ *is consistent. The weakening-based revision operator, denoted as* $\circ_{weak}$*, is defined as follows:*

$$\begin{aligned}O \circ_{weak} O' = \{&O_{weak} \cup \mathcal{T}' \cup \mathcal{A}' : O_{weak} \in Weak_{O'}(O) \ and \ \nexists O_i \in Weak_{O'}(O),\\ &d(O_i) < d(O_{weak})\}.\end{aligned}$$

**Example 4** *We consider an ontology* $O = \langle \mathcal{T}, \mathcal{A} \rangle = \langle \{Employee \sqsubseteq Person, Student \sqsubseteq Person, PhDStudent \sqsubseteq Student, Student \sqsubseteq \neg Employee\}, \{PhDStudent(paul)\} \rangle$.

*Now consider an axiom* $Employee(paul)$ *that is to be added to the ABox. The resulting ontology will be inconsistent. There are several alternatives to resolve the inconsistency be weakening either axioms in the* $\mathcal{T}$ *or the* $\mathcal{A}$*. For example,* $PhDStudent(paul)$ *might be weakened to* $Person(Paul)$*, the axiom* $PhDStudent \sqsubseteq Student$ *might be weakened to* $\top \sqsubseteq \neg PhDStudent \sqcup Person$ *which is equivalent to* $PhDStudent \sqsubseteq Person$ *(both with a degree of weakening of 1) to restore consistency.*

## 5.3 Discussion

In this section, we provide a comparison of our approach with the existing approaches according to the criteria proposed in Section 3.1.

- *Applications:* Our approach is proposed to deal with ontology revision. We consider not only debugging and repairing erroneous axioms in the TBox but also resolving inconsistency in the ABox and the TBox.
- *Granularity:* To resolve incoherence, the unit of change of our approach is an axiom, whilst when resolving inconsistency, the unit of change of our approach is either a concept or an individual.
- *Preservation of Structure:* We do not require to split the axioms. So our approach preserves the structure of the ontology.
- *Inconsistency vs. Incoherence:* Our approach deals with both inconsistency and incoherence in an ontology. Furthermore, we consider the interaction between inconsistency and incoherence.
- *Support for ABox, TBox:* We consider both incoherence in the TBox and inconsistency in the TBox and the ABox (or both).
- *Complexity:* Since our approach needs to resolve both incoherence and inconsistency, it is at least PSPACE-hard.
- *Support for Multiple/networked Ontologies:* Our approach can only deal with single, isolated ontologies.
- *Exploitation of context or background knowledge:* When resolving inconsistency in the TBox, our cardinality-maximizing consistent sub-ontology based approach may result in several consistent sub-ontologies, in this case, we need some context information to select an appropriate one. Therefore, our approach is dependent on context knowledge.
- *Interactivity, user involvement:* In many cases, we may need the user to made decision. For example, our approach for resolving inconsistency in the ABox needs to find out the axioms which are to be weakened. We can adapt and apply the debugging techniques to pinpoint axioms which are responsible for inconsistency, then we can ask users to select some axioms to weaken.
- *Availability of implementations:* We have implemented the cardinality-maximizing consistent sub-ontology based approach and debugging approach. The implementation will be reported in Chapter 6.

# Chapter 6

# Implementation

In this chapter, we describe the software implementation of our approach. We first give an overview of the reasoning services we have implemented in Section 6.1. These functionalities are accessible in two different ways in the RaDON system (Reasoning and Diagnosis in Ontology Networks) and the KAON2 OWL Tools: While the implementation of RaDON is based on extensions to the DIG interface, the KAON2 OWL Tools allow a command line based interface to the functionalities.

## 6.1 Implementation of Functionalities

The implemented functionalities basically support the individual steps of our approach to resolving inconsistency and incoherence presented in Figure 4.2. In particular, the implementation supports the following tasks of the process:

- *Consistency checking of TBox and ABox* Consistency checking is a standard reasoning task provided by DL reasoners. We simply perform separate checks for the TBox, ABox, and their union.
- *Checking the Coherence of TBox:* By definition, we simply need to check the coherence of all concepts of the ontology. This again is a standard reasoning task.
- *Inconsistency processing for TBox and ABox:* For the inconsistency processing, we have implemented the approach to return all cardinality-maximal subontologies.
- *Incoherence processing:* For each unsatisfiable concept, its minimal unsatisfiability-preserving sub-TBox (MUPs) will be given. In such case, we will point out all the minimal incoherence-preserving sub-TBox (MIPs) where the incoherence occurs.

## 6.2 RaDON - Implementation

RaDON is a system that extends the capabilities of existing reasoners with functionalities to deal with inconsistencies. These additional functionalities are made accessible via extensions to the DIG interface. The idea of extending the DIG interface [1] with non-standard reasoning services has been developed originally in the SEKT project and has for example been applied in PION [2] and evOWLution [3].

### 6.2.1 Architecture

The architecture of RaDON is shown in Figure 6.1 and consists of the following components:

[1] http://dl.kr.org/dig/
[2] http://wasp.cs.vu.nl/sekt/pion/
[3] http://evowlution.ontoware.org/

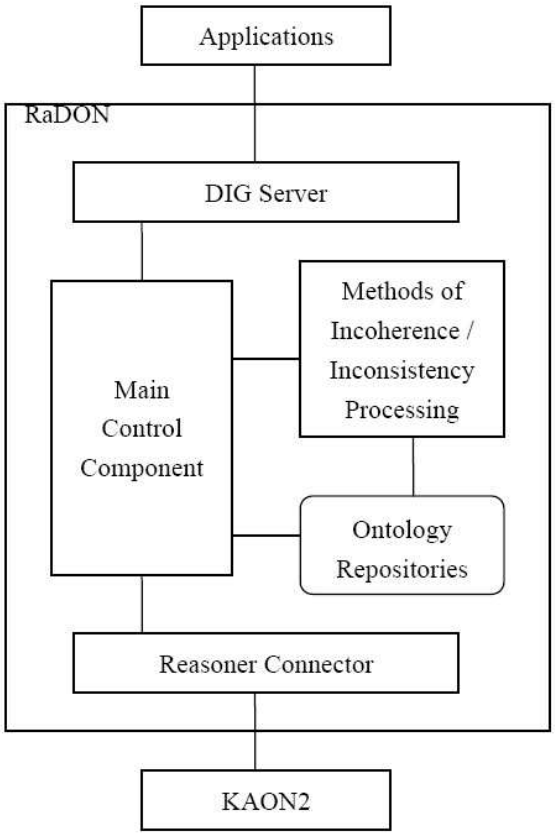


Figure 6.1: The Architecture of RaDON

- **DIG Server**: The XDIG server acts as RaDON's DIG server, which deals with requests from other ontology applications. It not only extends standard DIG request, like newKB request and tell request, but also provides additional reasoning facilities. Specifically, we extend standard newKB request by registering a default resolution function to detect the occurrence of logic inconsistency. The tell request is extended by proving the reason for inconsistency or incoherence to user.

- **Main Control Component**: The main control component performs the main processing to the methods for inconsistency processing and interact with the ontology repositories.

- **Methods of Inconsistency / Incoherence Processing**: This component provides various methods to deal with inconsistency or incoherence in the ontology. The methods include finding one maximal consistent subontology, or localizing inconsistency to find one minimal inconsistent subontology. Besides, some new methods are integrated into RaDON. For example, all of the cardinality-maximal consistent subontologies can be obtained. Another method is to calculate MUPs and MIPs to deal with incoherence.

- **Ontology Repositories**: The ontology repositories are used to store ontology statements which are provided by external ontology applications.

- **Reasoner Connector**: The connector is to invoke the KAON2 reasoner for reasoning tasks.

### 6.2.2 Download and Usage

The binary code of RaDON based on XDIG interface can be downloaded from the RaDON website `http://ontoware.org/projects/radon/`.
Unzip the RaDON package “radon.zip" into a directory, which is called `RaDON_ROOT` by us. The followings give a general introduction to the installation and usage with Windows System and jsdk1.5:

- **XDIG Server**: The XDIG server can be started by the following command under the directory of `RaDON_ROOT`:

  ```
  java -cp radon.jar;kaon2.jar
      org.semanticweb.kaon2.server.XServerMain -ontologies
      server_root -xdig -digport 8088
  ```

  The parameters in the command line are necessary for XDIG server. `-ontologies` is used to establish a local directory as a repository to store knowledge bases or ontologies. The value `server_root` for this parameter can be changed to any directory that you like. `-xdig` shows the XDIG server is desired to be started. For the port of XDIG server, we assign port 8088 to `-digport`.

- **Tomcat Server**: Copy the file “tell" in the directory `RaDON_ROOT/servlet` into the directory of Tomcat server, and make sure that the servlet with package “tell" will be mapped to `/servlet/TellConnector` by configuring `web.xml` file. So we can pass parameters and data obtained from demo webpage to the XDIG server by invoking the servlet-uri `http://localhost:8080/servlet/TellConnector`, which has been used in demo page. So if different servlet-uri is used, the servlet-uri in demo pages such as “left.htm" and `tellrequest.htm` should be configured as well.

- **Web page**: Open the `index.html` file in directory `RaDON_ROOT/www`, the demo menu can be found. The user needs to get a URI for the knowledge base or ontology before adding axioms to this knowledge base. In the demo page, user can simply use the button “create a new KB" on the left of the page to get a URI. For the approaches to dealing with inconsistency or incoherence, a default one to find a minimal inconsistent subontology is registered when a new knowledge base is created. The user can also choose different approach on the left of the demo page. On the right of this page, tell request serves to add axioms to a knowledge base.

### 6.2.3 Examples

Here is an example in DIG format, which describes a consistent ontology $O_1$:

```
<impliesc>
     <catom name="Employee"/>
     <catom name="Person"/>
</impliesc>
<impliesc>
     <catom name="Student"/>
     <catom name="Person"/>
</impliesc>
<impliesc>
     <catom name="PhdStudent"/>
     <catom name="Student"/>
</impliesc>
<disjoint>
```

```
     <catom name="Student"/>
     <catom name="Employee"/>
</disjoint>
<instanceof>
     <individual name="John"/>
     <catom name="PhdStudent"/>
</instanceof>
```

Now consider a new ontology $O_2$ which includes one axiom:

```
<instanceof>
    <individual name="John"/>
    <catom name="Employee"/>
</instanceof>
```

The resulting ontology $O_1 \sqcup O_2$ is coherent, but inconsistent. There exist a number of cardinality-maximizing consistent sub-ontologies that can be obtained by removing either one of the following axioms from $O_1$: $PhDStudent \sqsubseteq Student$, $Student \sqsubseteq \neg Employee$, $john \in PhdStudent$. Thus, the response when adding $O_2$ to $O_1$ is shown as follows:

```
<response>
   <error message="Inconsistent ontology"/>
      The cardinality-maximal consistent subsets include:
      <sub>
         [subClassOf Employee Person]
         [subClassOf Student Person]
         [classMember Employee John]
         [disjoint Student Employee]
         [classMember PhdStudent John]
      </sub>
      <sub>
         [subClassOf Employee Person]
         [subClassOf Student Person]
         [classMember Employee John]
         [disjoint Student Employee]
         [subClassOf PhdStudent Student]
      </sub>
      <sub>
         [subClassOf Employee Person]
         [subClassOf Student Person]
         [classMember Employee John]
         [classMember PhdStudent John]
         [subClassOf PhdStudent Student]
      </sub>
   </error>
</response>
```

## 6.3 Extensions to the KAON2 OWL Tools

### 6.3.1 Architecture of KAON2 OWL Tools

The OWL tools [4] are a set of tools for working on OWL files, exposing the abilities of the KAON2 ontology infrastructure to the command line. We integrate the functionalities in RaDON into OWL tools to facilitate the task of the users who are used to using OWL tools or who prefer to use the command line for the test.

### 6.3.2 Download and Usage

The binary version (`owltools-radon.jar`) and source code of RaDON with OWL tools can be downloaded from the website of RaDON. We provide different parameters for the approaches to dealing with inconsistency or incoherence. For example, the following command shows the method to get all cardinality-maximal consistent subontologies to be returned if inconsistency occurs when adding one axiom in onto2.owl to onto1.owl and onto1.owl is consistent.

```
java -cp kaon2.jar;owltools-radon.jar
edu.unika.aifb.owltools.OWLTools radon onto1.owl onto2.owl -cardi
```

For the parameters, such as `-mip`, `-mup` or `-mupmip`, only one ontology is needed. If there are still two ontologies, we will check the incoherence by combining the two ontologies together. For each test, only one approach to dealing with inconsistency or incoherence can be selected. And the first declared approach will be selected if more than one approaches are declared.

### 6.3.3 Example

We test the example `buggyPolicy.owl` to show all the MUPs which are related to the unsatisfiable concepts in the ontology and all the MIPs. This example can be downloaded from the website `http://www.mindswap.org/2005/debugging/ontologies/`. For the command line, we just input the following command:

```
java -cp kaon2.jar;owltools-radon.jar
 edu.unika.aifb.owltools.OWLTools radon buggyPolicy.owl -mupmip
```

The part of the results are shown in figure 6.2.

[4] http://owltools.ontoware.org/

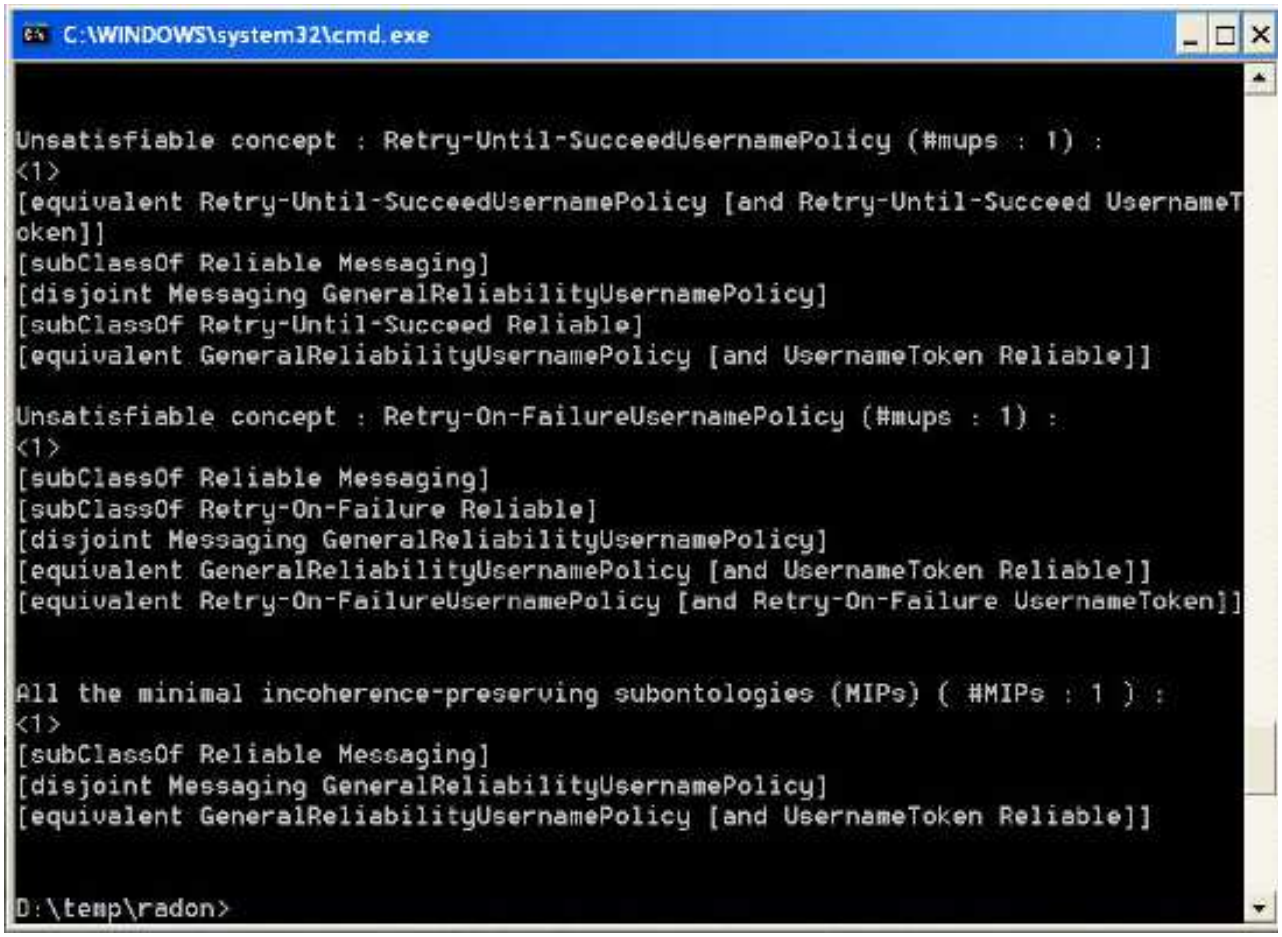


Figure 6.2: The results of RaDON with OWL tools

# Chapter 7

# Conclusion

## 7.1 Summary

In this deliverable we addressed the question of how to define appropriate models for consistency in evolving networks of ontologies. We first proposed a set of evaluation criteria to facilitate the comparison of existing approaches to dealing with inconsistencies in evolving ontologies. We then proposed a novel general approach for resolving inconsistency and incoherence in ontologies. We instantiated our approach by proposing a concrete approach to resolving incoherence and some concrete approaches to resolving inconsistency. By comparing our approach with existing approaches with respect to the criteria, we show that our approach is powerful enough to be used to deal with inconsistency and incoherence in evolving ontologies. Finally, we provided implementations of the proposed methods.

## 7.2 Roadmap

There are various dimensions of future work to be considered in the scope of the NeOn project.

In the subsequent deliverable, in *D1.2.2 Consistency Models for Networked Ontologies - Evaluation* we will build on the work presented in this deliverable and evaluate our approaches using real life data. Ideally, this data will be directly obtained from the NeOn case studies.

Further, we will continue the developments of our approaches for consistency models, both on the conceptual and on the implementation level. On the conceptual level, we might consider other approaches to dealing with inconsistencies, especially with respect to consistency models for networked ontologies. In alignment with the work performed in WP3 on context, we will consider how we can exploit context information in order to deal with inconsistencies.

On the implementation level, we will make our RaDON system compatible with new upcoming standards, such as OWL 1.1 and DIG 2.0. While in the current implementation, the functionalities are accessible as proprietary extensions to the DIG1.1 interface, the DIG2.0 interface will allow to define extensions – such as the ones for our non-standard reasoning services – in a generic way.

Further we will work on the integration of our implementations into the NeOn architecture. This integration can be performed both in terms of a plugin for the NeOn toolkit (e.g. as a diagnosis tool for networked ontologies for the end user) as well as backend services within the NeOn infrastructure as extensions to the NeOn reasoners.

# Bibliography

[BCM+03] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):3443, 2001.

[FFKI06] Marcelo A. Falappa, Eduardo L. Fermé, and Gabriele Kern-Isberner. On the logic of theory change: Relations between incision and selection functions. In *Proc. of ECAI'06*, pages 402–406, 2006.

[FHP+06] Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache. Inconsistencies, negations and changes in ontologies. In *Proc. of AAAI'06*, 2006.

[FPA04] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. Generalizing the agm postulates: preliminary results and applications. In *NMR*, pages 171–179, 2004.

[FPA05] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. On applying the AGM theory to DLs and OWL. In *Proc. of 4th International Conference on Semantic Web (ISWC'05)*, pages 216–231. 2005.

[FPA06] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. Evolving ontology evolution. In *SOFSEM*, pages 14–29, 2006.

[FS05] Gerhard Friedrich and Kostyantyn M. Shchekotykhin. A general diagnosis method for ontologies. In *Proc. of 4th International Conference on Semantic Web (ISWC'05)*, pages 232–246, 2005.

[Fuh91] A. Fuhrmann. Theory contraction through base contraction. *Journal of Philosophical Logic*, 20:175–203, 1991.

[Gar88] Peter Gardenfors. *Knowledge in Flux-Modeling the Dynamic of Epistemic States*. The MIT Press, Cambridge, Mass, 1988.

[Gin86] Matthew L. Ginsberg. Counterfactuals. *Artif. Intell.*, 30(1):35–79, 1986.

[Han94] Sven Ove Hansson. Kernel contraction. *J. Symb. Log.*, 59(3):845–859, 1994.

[HM01] Volker Haarslev and Ralf Möller. RACER system description. In *IJCAR'01*, pages 701–706, 2001.

[Hor98] Ian Horrocks. The fact system. In *Proc. of International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*, pages 307–312. Springer, 1998.

[HvHH+05] Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, and York Sure:. A framework for handling inconsistency in changing ontologies. In *Proc. of 4th International Semantic Web Conference (ISWC'05)*, pages 353–367. Springer, 2005.

[HvHtT05] Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with inconsistent ontologies. In *Proc. of 19th International Joint Conference on Artificial Intelligence(IJCAI'05)*, pages 254–259. Morgan Kaufmann, 2005.

[KM92] Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artif. Intell.*, 52(3):263–294, 1992.

[KPGS06] Aditya Kalyanpur, Bijan Parsia, Bernardo Cuenca Grau, and Evren Sirin. Justifications for entailments in expressive description logics. Technical report, University of Maryland Institute for Advanced Computer Studies (UMIACS), 2006.

[KPSG06] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca Grau. Repairing unsatisfiable concepts in owl ontologies. In *ESWC'06*, pages 170–184, 2006.

[LPSV06] Joey Lam, Jeff Z. Pan, Derek Seeman, and Wamberto Vasconcelos. A fine-grained approach to resolving unsatisfiable ontologies. In *Proc. of WI'06*, 2006.

[MLB05] Thomas Meyer, Kevin Lee, and Richard Booth. Knowledge integration for description logics. In *Proc. of 20th National Conference on Artificial Intelligence (AAAI'05)*, pages 645–650. AAAI Press, 2005.

[MLBP06] Thomas Meyer, Kevin Lee, Richard Booth, and Jeff Z. Pan. Finding maximally satisfiable terminologies for the description logic alc. In *Proc. of AAAI'06*, 2006.

[Neb90] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artif. Intell.*, 43(2):235–249, 1990.

[PSK05] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL ontologies. In *Proc. of WWW'05*, pages 633–640, 2005.

[PT06] Peter Plessers and Olga De Troyer. Resolving inconsistencies in evolving ontologies. In *Proc. of ESWC'06*, pages 200–214, 2006.

[QLB06a] Guilin Qi, Weiru Liu, and David A. Bell. Knowledge base revision in description logics. In *Proc. of JELIA'06*, pages 386–398. Springer Verlag, 2006.

[QLB06b] Guilin Qi, Weiru Liu, and David A. Bell. A revision-based algorithm for handling inconsistency in description logics. In *Proc. of NMR'06*, pages 124–132, 2006.

[Rei87] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[SC03] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI'03*, pages 355–362, 2003.

[Sch05] Stefan Schlobach. Diagnosing terminologies. In *Proc. of AAAI'05*, pages 670–675, 2005.

[SHC06] Stefan Schlobach, Zhisheng Huang, and Ronald Cornet. Inconsistent ontology diagnosis: Evaluation. Technical report, Department of Artificial Intelligence, Vrije University Amsterdam; SEKT Deliverable D3.6.2, 2006.

[SS04] S. Staab and R. Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.