An Analysis of Approaches to Resolving Inconsistencies in DL-based Ontologies

Peter Haase, Guilin Qi

Institute AIFB, University of Karlsruhe,
D-76128 Karlsruhe, Germany
{haase, gqi}@aifb.uni-karlsruhe.de

Abstract. In this paper we provide an overview and analysis of approaches for dealing with inconsistencies in DL-based ontologies. We propose criteria for the comparison of the different approaches. These criteria facilitate the users to choose an appropriate approach to dealing with inconsistencies for their purpose.

1 Introduction

The problem of inconsistency (or incoherence) handling in ontologies is recently attracting a lot of attention. Inconsistency can occur due to several reasons, such as modeling errors, migration from other formalisms, merging ontologies, and ontology evolution.

One way to deal with logical contradictions is to resolve logical modeling errors whenever a logical problem is encountered[1]. Several methods have been proposed to debug erroneous terminologies and have them repaired when inconsistencies are detected [SC03,Sch05,PSK05,FS05].

Considering the varieties of approaches to resolving inconsistency, we require criteria for comparing these approaches to facilitate the selection of an appropriate approach. Three criteria are proposed in [HvHH+05]: (1) is the approach applied at development or at runtime, (2) is the input a consistent or an inconsistent ontology, (3) is the output a consistent or an inconsistent ontology (or an answer). While these criteria already provide a useful classification, they are not sufficient for the selection of an adequate approach for a given application scenario.

In this paper, we first propose some criteria for comparing approaches for dealing with inconsistencies. We then give an overview of approaches for resolving inconsistency and compare them against the proposed criteria.

The paper is organized as follows. Section 2 provides some basic notions of terminology debugging. Some evaluation criteria are proposed in Section 3. We then give an overview of approaches for repairing inconsistency in Section 4. In Section 5, we compare existing approaches against the proposed criteria. Finally, we conclude the paper in Section 6.

[1] Another way to deal with inconsistencies is to reason with inconsistent ontologies (such as the approach given in [HvHtT05]). However, the analysis of them is out of the scope of this paper.

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 this paper, 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. 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.*

Let us consider an ontology $O = \{C_1(a), C_2(b), C_3 \sqsubseteq C_1, C_3 \sqsubseteq C_2, C_1 \sqsubseteq \neq C_2\}$. It is clear that O is incoherent because C_3 is an unsatisfiable concept in O and it is consistent. 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, typically they do not support the diagnosis and incoherence resolution. 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:

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

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:

$mips(\mathcal{T}) = \{\{ax_1, ax_2\}, \{ax_3, ax_4, ax_5\}, \{ax_4, ax_7\}\}$

3 Criteria

In this section we discuss a number of criteria for the comparison of approaches to dealing with inconsistency in distributed knowledge bases.

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.Depending on the assumptions, particular approaches may be applicable to one or many of these application scenarios.

Granularity of Repair Dealing with and resolving inconsistencies can be performed on different levels of granularity. Most approaches consider a knowledge base as a set of axioms. In a trivial way, knowledge bases may be repaired by simply removing problematic axioms completely. However, often only parts of an axiom may be the true cause of an inconsistency. In such cases, more fine granular approaches (e.g. on the level of concepts) would be desirable. Some 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. If a repair is performed based on that normal form, the effect on the original structure of the knowledge base may be hard to reconstruct. However, the traceability of effects in terms of the original knowledge base is often important for 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 terminological and assertional knowledge When dealing with inconsistencies it often is important to consider whether an inconsistency occurs in the TBox (as part of the terminological knowledge) or whether the ABox (assertional knowledge, i.e. the data corresponding to some TBox) is inconsistent. 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. Some approaches provide solutions to dealing with either one type of inconsistency and disregard the other. Others do not make the distinction at all.

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. Complexity results

thus depend on the complexity on the supported logic on the one hand, and the properties of the algorithms for handling inconsistencies on the other hand. In order to assure practicability, these complexity issues need to be taken into account.

Support for Multiple/networked knowledge bases 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. Other approaches may be applied to multiple knowledge base scenarios by simply considering the union of the individual knowledge bases as a single knowledge base. We evaluate what kind of networking relationships are supported or how the approach can be extended to operate with multiple ontologies.

Exploitation of background / context 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.

4 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 repairing incoherent ontologies. The first group comes from the Vrije Universiteit Amsterdam and the second group is the MindSwap 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.

4.1 Debugging and Diagnose DL-based Ontologies in MUPSter

We recapitulate 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 tableaux from a branch B initially containing only *labelled formula* $(a \ : \ A)^{\emptyset}$ (for a new individual name a) by applying the tableaux rules as long as possible. The rules are standard $\mathcal{ALC}$-tableaux rules with lazy unfolding, and have to be read as follows: assume that there is a tableaux $T = \{B, B1, ..., B_n\}$ with $n+1$ branches. After applying one of the rules on B, we get a tableaux $T' = \{B', B_1, ..., B_n\}$ 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 minimization function for A and $\mathcal{T}$ according to the tableaux rules. A propositional formula ϕ is called a *minimization 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 ϕ true. Here axioms are used as propositional variable in ϕ. As we can identify unsatisfiability of A *w.r.t* a set S of axioms with a closed tableaux 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 ϕ, 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_σ joined to n by an edge labeled by σ. 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 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).

4.2 Debugging and Repairing OWL Ontologies in SWOOP

A drawback of the debugging approach in [SC03] is that it is restricted to unfoldable ALC TBoxes. Furthermore, it is based on the tableaux algorithms for DLs. Therefore, it is dependent on the tableaux reasoner. In [PSK05,KPGS06], two

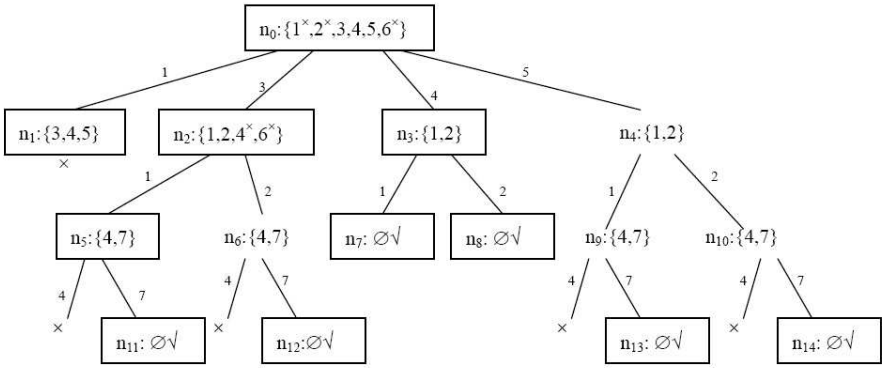

Fig. 1. HS-Tree with small conflict sets

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 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.

4.3 Consistent Ontology Evolution

[HS05] describes a process to support the consistent evolution of OWL DL based ontologies, which ensures that the consistency of an ontology is preserved when changes are applied to the ontology. The process consists of two main phases: (1) *Inconsistency Detection*, which is responsible for checking the consistency of an ontology with the respect to the ontology consistency definition and identifying parts in the ontology that do not meet consistency conditions; (2) *Change Generation*, which is responsible for ensuring the consistency of the ontology by generating additional changes that resolve detected inconsistencies. The authors define methods for detecting and resolving inconsistencies in an OWL ontology

after the application of a change. As for some changes there may be several different consistent states of the ontology, *resolution strategies* allow the user to control the evolution.

The methods for detecting inconsistencies rely on the idea of a selection function are to identify the *relevant* axioms that contribute to the inconsistency. In the most simple case, syntactic relevance – considering how the axioms of the ontology are structurally connected with the change – is used. Based the selection function, algorithms to find *minimal inconsistent subontologies* and *maximal consistent subontologies* are presented.

The approach only supports repairs by removing complete axioms from the ontology, a weakening based on a finer granularity as future extension is mentioned, but no algorithms are proposed. The approach does not make a distinction between ABox and TBox axioms, as such both ABox and TBox inconsistencies are trivially supported. Further, the approach does not provide any explicit support for dealing with networked ontologies.

4.4 AGM's Postulates for Belief Change 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. Their work is based on the *coherence model*[2]. 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. In [FHP$^+$06], the authors first defined the notions of axiom negation and discussed postulates for revision. However, explicit construction of a revision operator was not considered in these papers.

4.5 Knowledge Base Revision in Description Logics

The work in [FPA04,FPA05,FPA06] presumes that the original DL knowledge base is closed under logical consequence relation and the result of revision is

[2] The notion of coherence model has nothing to do the notion of incoherence in DLs defined in Section 2.

still a DL knowledge base which is closed under logical consequence relation. In [QLB06a], the revised AGM postulates for belief revision in [KM92] were generalized and two revision operators which satisfy the generalized postulates were given. 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.

4.6 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.

5 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 1 and Table 2.

According to Table 1 and Table 2, MUPSter and SWOOP are mainly applied to debug and repair incoherence, whilst other approaches are applied to deal with inconsistency. When resolving incoherence, MUPSter will delete axioms in the TBox and SWOOP deletes either axioms or concepts in the TBox. To resolve inconsistency, the knowledge base revision approaches and knowledge integration approaches either delete axioms in a DL knowledge base or remove some instances which are responsible for inconsistency. However, the AGM-based approach for revision in DLs and the approach on consistent ontology evolution simply delete the whole axioms. Before dealing with incoherence, both MUPster and SWOOP may split the axioms into smaller axioms, so the structure will be lost. One of the knowledge base revision approaches given in [QLB06a], called

Table 1. Evaluation results

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

Table 2. Evaluation results

Criteria	consistent ontology evolution	knowledge base revision	knowledge integration
Application	ontology evolution	revision	merging
Granularity	axiom	axiom or instance	axiom or instance
Preservation of structure	yes	partially	partially
Support for ABox, TBox	TBox and ABox	TBox and ABox	TBox and ABox
Inconsistency vs. Incoherence	inconsistency	inconsistency	inconsistency
Complexity	PSPACE-hard	PSPACE-hard	PSPACE-hard
User involvement	yes	no	no
Availability of implementation	yes	no	no
Support for networked ontologies	no	no	yes
Exploitation of context	no	no	yes

refined weakening-based revision approach, also requires to split axioms, so it does not preserve the structure of the axioms. The weakening-based revision approach in [QLB06a] does not change 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. Debugging in MUPSTter requires satisfiability checking in the DL $\mathcal{ALC}$, which is known to be EXPTIME-complete. Depending on the supported underlying DL, the other approaches are at least PSPACE-hard. Among all the approaches, only MUPSter and SWOOP are implemented, and only SWOOP has a user interface. The MUPSter, AGM-based approaches and the knowledge base revision approaches do not exploit context information to deal with incoherence or inconsistency, whilst SWOOP and knowledge integration approaches exploit ranking information to resolve incoherence or inconsistency.

6 Conclusion

In this paper, we have provided a survey of existing work on resolving inconsistency in DL-based ontologies. We then proposed a set of criteria for comparing between different approaches.

It does not come as a surprise that none of the surveyed approaches is universally applicable for any application scenario, instead different approaches are good for different purposes. Our comparison aims at supporting the selection of the right tools for the job.

As a general observation we see that the use of context information in the process of repairing ontologies as well as the support for multiple, networked ontologies is still underdeveloped. As part of our future work, we plan to advance the techniques for dealing with inconsistencies in these directions.

Acknowledgements Research presented in this paper was supported by the European Commission under contract IST-2006-027595 NeOn (http://www.neon-project.org/).

References

[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.

[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.

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

[HS05] Peter Haase and Ljiljana Stojanovic. Consistent evolution of owl ontologies. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *ESWC*, volume 3532 of *Lecture Notes in Computer Science*, pages 182–197. Springer, 2005.

[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.

[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.

[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.

[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.