

A Fine-Grained Approach to Resolving Unsatisfiable Ontologies

Sik Chun (Joey) Lam, Jeff Z. Pan, Derek Sleeman, Wamberto Vasconcelos
 Department of Computing Science
 University of Aberdeen, AB24 3UE, UK
 {slam, jpan, sleeman, wvasconc}@csd.abdn.ac.uk

Abstract

In the Semantic Web, inconsistencies in OWL ontologies may easily occur. Existing approaches either identify the minimally unsatisfiable sub-ontologies or calculate the maximally satisfiable sub-ontologies. However practical problems remain; it is not clear which axioms or which parts of axioms should be selected for repair, and how to repair those axioms. In this paper, we address this limitation by proposing a fine-grained approach to resolving unsatisfiable ontologies. We revise the axiom tracing technique first proposed by Baader and Hollunder, so as to track which parts of the problematic axioms cause the unsatisfiability. Moreover, we support ontology users in rewriting problematic axioms. In order to minimise the impact of changes and prevent unintended entailment loss, harmful and helpful changes are identified and provided as guidelines. Based on the methods described we present a preliminary version of an interactive debugging tool and demonstrate its applicability in practice.

1 Introduction

An increasing number of OWL ontologies are appearing on the Semantic Web, and ontology languages are becoming more expressive. However, this makes resolving inconsistencies in ontologies a challenging task for ontology modelers. Standard Description Logic (DL) [2] reasoning services can check if an ontology is satisfiable; however, they do not provide support for resolving unsatisfiability. Other existing approaches mainly focus on how to identify problematic axioms (by providing the minimally unsatisfiable sub-ontologies) [5] or how to directly weaken the target unsatisfiable ontology (by providing all the possible maximally satisfiable sub-ontologies) [4]. However practical problems remain; it is not clear which axioms or which parts of axioms should be selected for repair, and how to repair those axioms.

Example 1 *Assume that an ontology \mathcal{O} contains the following axioms:*

$$\alpha_1: A \doteq C \sqcap D \sqcap G, \quad \alpha_2: C \doteq E \sqcap F,$$

$$\alpha_3: E \doteq \neg D \sqcap (\exists R.D), \quad \alpha_4: H \doteq \forall R.C.$$

It can be shown that the concept A is unsatisfiable, by using standard DL TBox reasoning. Existing approaches either identify the minimally unsatisfiable sub-ontology $\mathcal{O}_{min} = \{\alpha_1, \alpha_2, \alpha_3\}$ or calculate the maximally satisfiable sub-ontologies $\mathcal{O}_{max1} = \{\alpha_2, \alpha_3, \alpha_4\}$, $\mathcal{O}_{max2} = \{\alpha_1, \alpha_3, \alpha_4\}$ and $\mathcal{O}_{max3} = \{\alpha_1, \alpha_2, \alpha_4\}$. In short, one of the axioms α_1 , α_2 or α_3 should be removed from \mathcal{O} . However, it is easy to show that we do not need to remove any of the above ‘whole’ axioms. In fact, we can simply remove any one of the following parts of axioms: (a) $A \sqsubseteq C$, (b) $A \sqsubseteq D$, (c) $C \sqsubseteq E$, or (d) $E \sqsubseteq \neg D$, and \mathcal{O} becomes satisfiable.

In this paper, we propose an algorithm that extends Meyer et al.’s tableaux algorithm [4] (the kind of tracing technique was first proposed by Baader and Hollunder [1]). The algorithm not only pinpoints the problematic axioms, but also traces which parts of the axioms are responsible for the unsatisfiability of the target concept A . Using this algorithm, we make the following two contributions. The first is to calculate the lost entailments of named concepts due to the removal of axioms. Whenever parts of an axiom or a whole axiom is removed, it frequently happens that intended entailments are lost. In order to minimise the impact on the ontology, we analyse the lost entailments (e.g. concept subsumption) of named concepts due to the removal of (parts of) axioms. The second contribution is to identify harmful and helpful changes; this is where the fine-grained tracing information is useful to facilitate rewriting the problematic axioms, rather than removing them completely. It should be noted that improperly rewriting a problematic axiom might not resolve the unsatisfiability, and could introduce additional unsatisfiability. For this purpose we define *harmful* and *helpful* changes with respect to a concept in a problematic axiom. A harmful change cannot resolve the

problem, and might cause additional unsatisfiability in the ontology; a helpful change will resolve the problem without causing additional contradictions, and will compensate for the lost entailments. Tools based on such techniques would help users to resolve unsatisfiable ontologies. To evaluate this vision, we implement a preliminary version of an interactive debugging tool and demonstrate its applicability in practice.

The rest of this paper is organised as follows. Section 2 briefly introduces ontologies and the Description Logic \mathcal{ALC} . Section 3 presents our approach to supporting the rewriting of problematic axioms. Section 4 presents the preliminary evaluation. The paper closes with a discussion of related work and final conclusions.

2 Ontology and the \mathcal{ALC} DL

An ontology typically consists of a hierarchical description of important concepts in a domain, along with descriptions of the properties of each concept, and constraints on these concepts and properties. Description Logics (DLs) are a class of knowledge representation languages to represent and reason about ontologies. In this paper, we consider \mathcal{ALC} Description Logic [7] based ontologies to illustrate our approach.

Let N_C and N_R be a set of concept names and role names respectively; an ontology \mathcal{O} consists of a set of *terminology* axioms \mathcal{T} (TBox) and assertional axioms \mathcal{A} (ABox). An axiom in \mathcal{T} is either of the form $C \sqsubseteq D$ or $C \doteq D$, where C and D are arbitrary concepts; an axiom in \mathcal{A} is either of the form $a : C$ (where C is a concept and a is an individual name; a belongs to C), or of the form $\langle a, b \rangle : R$ (where a, b are individual names and R is a role name; b is a filler of the role R for a). A Tbox \mathcal{T} is *unfoldable* iff the left-hand side of every $\alpha \in \mathcal{T}$ contains a concept name A , there are no other α 's with A on the left-hand side, and the right-hand side of α contains no direct or indirect references to A . An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set of individuals (the domain of the interpretation) and an interpretation function $(\cdot^{\mathcal{I}})$, which maps each concept name $CN \in N_C$ to a set $CN^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role name $RN \in N_R$ to a binary relation $RN^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. An interpretation \mathcal{I} *satisfies* a concept inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation \mathcal{I} *satisfies* a concept assertion $a : C$ (a role assertion $\langle a, b \rangle : R$) if $a \in C^{\mathcal{I}}$ ($\langle a, b \rangle \in R^{\mathcal{I}}$, respectively).

3 Proposed Approach

We revise the specialised tableau-based algorithm from Meyer et al.[4] (the kind of tracing technique was first proposed by Baader and Hollunder [1]). Instead of excluding complete axioms involved in a clash, our algorithm captures the concept components of axioms

responsible for the clash. As a result, we are able to trace sequences of concept components of axioms w.r.t a contradiction in the ontology.

We assume that $\mathcal{T} = \{\alpha_1, \dots, \alpha_n\}$, with α_i referring to $A_i \doteq C_i$ or $A_i \sqsubseteq C_i$ for $i = 1, \dots, n$. The tableau-based algorithm decides the satisfiability of A_i w.r.t \mathcal{T} by trying to construct a model for it, called a tree \mathbf{T} . Each node x in the tree is labeled with a set $\mathcal{L}(x)$ containing elements of the form $(a : C, I, a' : C')$, where C and C' are concepts, a and a' are individual names, and I is an index-set. This means that the individual a belongs to concept C due to an application of an expansion rule on $a' : C'$; the set of axioms, in which $a : C$ comes from, is recorded in the index-set. This is done by adding i to I , which is a set of integers in the range $1, \dots, n$. In an element of the form $(a : C, I, a' : C')$ we frequently refer to C as “the concept”, and a as “the individual” (i.e. we are referring to the first concept assertion).

Table 1. Tableau expansion rules for \mathcal{ALC}

U_{\doteq}^+ -rule	if $A_i \doteq C_i \in \mathcal{T}$, and ($a : A_i, I, a' : A'$) $\in \mathcal{L}(x)$ and is not tagged, then $\text{tag}((a : A_i, I, a' : A'))$ and $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : C_i, I \cup \{i\}, a' : A')\}$
U_{\sqsubseteq}^- -rule	if $A_i \sqsubseteq C_i \in \mathcal{T}$, and ($a : \neg A_i, I, a' : A'$) $\in \mathcal{L}(x)$ and is not tagged, then $\text{tag}((a : \neg A_i, I, a' : A'))$ and $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : \neg C_i, I \cup \{i\}, a' : A')\}$
U_{\sqsubseteq}^- -rule	if $A_i \sqsubseteq C_i \in \mathcal{T}$, and ($a : A_i, I, a' : A'$) $\in \mathcal{L}(x)$ and is not tagged, then $\text{tag}((a : A_i, I, a' : A'))$ and $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : C_i, I \cup \{i\}, a' : A')\}$
\sqcap -rule	if $(a : C_1 \sqcap C_2, I, a' : A') \in \mathcal{L}(x)$, then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : C_1, I, a : C_1 \sqcap C_2)\} \cup$ $\{(a : C_2, I, a : C_1 \sqcap C_2)\}$
\sqcup -rule	if $(a : C_1 \sqcup C_2, I, a' : A') \in \mathcal{L}(x)$, then $\mathcal{L}(y) := \mathcal{L}(x) \cup \{(a : C_1, I, a : C_1 \sqcup C_2)\}$ and $\mathcal{L}(z) := \mathcal{L}(x) \cup \{(a : C_2, I, a : C_1 \sqcup C_2)\}$
\exists -rule	if $(a : \exists R.C, I, a' : A') \in \mathcal{L}(x)$, and the above rules cannot be applied, then $X := \{(b : C, I, a : \exists R.C)\} \cup$ $\{(b : D, I \cup J, a : \forall R.D)\} \cup \{(a : \forall R.D, J, a' : A'_2)\}$ $\in \mathcal{L}(x)$ $\mathcal{L}(x) := \mathcal{L}(x) \cup X$, where b is a new individual name not occurring in $\mathcal{L}(x)$

To determine the satisfiability of a concept A_i in \mathcal{T} for some $i = 1, \dots, n$, a tree \mathbf{T} is initialised to contain a single node x , with $\mathcal{L}(x) = \{(a : A_i, \emptyset, nil)\}$. During the expansion, C_i is assumed to be converted to negation normal form. We repeatedly expand the tree by applying the rules in Table 1 as many times as possible. A node is fully expanded when none of the rules can be applied to it. \mathbf{T} is fully expanded when all of its leaf nodes are fully expanded. \mathbf{T} contains an obvious *clash* when, for some node x , some individual b and some concept C , $\{(b : C, -, -), (b : \neg C, -, -)\} \subseteq \mathcal{L}(x)$.¹

¹“-” stands for any value, that is, it is a place holder.

When a clash is found, the classical tableaux algorithm [1] will either backtrack and select a different leaf node (which is created by applying the \sqcup -rule), or report the clash and terminate, if no node remains to be expanded. Because the rules are still applicable to a node even when a clash is found, there may be more than one clash in the node, while this clash may also occur in other nodes (repeated nodes). As a result, we can obtain all the clashes in the tree and eliminate the repeated clashes. If the input of the tableaux algorithm is a concept A and terminology \mathcal{T} , we have the following three properties²: (1) the algorithm always terminates; (2) A is unsatisfiable iff all leaf nodes in the tree \mathbf{T} contain at least one clash; (3) an unsatisfiable concept will become satisfiable if all the clashes in any one of the leaf nodes are resolved. This is because whenever the non-deterministic \sqcup -rule is applied, two new leaf nodes are created. This is the only way to create the leaf nodes. This implies that if either of the two leaf nodes has no clash, the concept is satisfiable, so we only have to resolve the clashes in one of the leaf nodes.

Figure 1. Application of tableaux expansion rules on Example 1

- (1) Initialise the root node x with $\mathcal{L}(x) := \{(a : A, \emptyset, nil)\}$,
- (2) Apply the C^+ -rule to $(a : A, \emptyset, nil)$,
it gives $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : C \sqcap D \sqcap G, \{1\}, a : A)\}$,
- (3) Apply twice the \sqcap -rule to $(a : C \sqcap D \sqcap G, \{1\}, a : A)$,
it gives $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : C, \{1\}, a : C \sqcap D \sqcap G),$
 $(a : D, \{1\}, a : C \sqcap D \sqcap G), (a : G, \{1\}, a : C \sqcap D \sqcap G)\}$,
- (4) Apply the C^+ -rule to $(a : C, \{1\}, a : C \sqcap D \sqcap G)$,
followed by applying the \sqcap -rule,
it gives $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : E \sqcap F, \{1, 2\}, a : C),$
 $(a : E, \{1, 2\}, a : E \sqcap F), (a : F, \{1, 2\}, a : E \sqcap F)\}$,
- (5) Apply the C^+ -rule to $(a : E, \{1, 2\}, a : C)$,
it gives $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : \neg D \sqcap \exists R.D, \{1, 2, 3\}, a : E)\}$;
- (6) Apply the \sqcap -rule to $(a : \neg D \sqcap \exists R.D, \{1, 2, 3\}, a : E)$,
it gives $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : \neg D, \{1, 2, 3\}, a : \neg D \sqcap \exists R.D),$
 $(a : \exists R.D, \{1, 2, 3\}, a : \neg D \sqcap \exists R.D)\}$;
- (7) Apply the \exists -rule to $(a : \exists R.D, \{1, 2, 3\}, a : \neg D \sqcap \exists R.D)$,
it gives $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(b : D, \{1, 2, 3\}, a : \exists R.D)\}$.

Figure 1 shows how the tableaux algorithm is applied on Example 1³ to check for the unsatisfiability of A . The tree \mathbf{T} contains a node x which has a clash

²The proof of termination, soundness and completeness for the original tableau algorithm can be found in Baader et al. [1].

³For simplicity, the process of tagging concepts is not shown.

because $\{(a : D, \{1\}, a : C \sqcap D \sqcap G),$
 $(a : \neg D, \{1, 2, 3\}, a : \neg D \sqcap \exists R.D)\} \subseteq \mathcal{L}(x)$.

Definition 1 (Sequences of a Clash) *Given a clash in a tree, the sequences of a clash, Seq^+ and Seq^- , contain elements involved in the clash. The sequences are of the form $\langle (a_0 : C_0, I_0, a_1 : C_1), (a_1 : C_1, I_1, a_2 : C_2), \dots, (a_{n-1} : C_{n-1}, I_{n-1}, a_n : C_n), (a_n : C_n, \emptyset, nil) \rangle$, where $I_{i-1} \subseteq I_i$ for each $i = 1, \dots, n$. The first elements of Seq^+ and Seq^- are of the form $(a : C, I', a' : C')$ and $(a : \neg C, I'', a'' : C'')$ respectively. The last element of both sequences is $(a_n : C_n, \emptyset, nil)$, where C_n is the unsatisfiable concept.*

Let there be a fully expanded tree \mathbf{T} for some unsatisfiable concept, for which we can detect a set of clashes: from each clash we can obtain two sequences of elements corresponding to the clash (see Definition 1). In our example, A is checked to be unsatisfiable, the sequences of elements involved in the clash can be obtained (see Figure 2). Note that the union of the index sets of the first element in the two sequences gives the set of axioms which cause A to be unsatisfiable. This set of axioms is the same as the notion of minimal unsatisfiability preserving sub-TBoxes (MUPS) in [6]. It is obvious that the concepts in the elements of the sequences trigger a clash, we therefore know which concepts in axioms are relevant to the clash. Such concepts in axioms are annotated with a specific superscript number corresponding to a clash. Note that a concept may be involved in more than one clash, therefore it may be annotated with more than one number. We introduce the notion of arity of a concept in an axiom, denoted by $arity(\alpha, C)$, to count the number of times it appears in the clashes. This idea is similar to the core of MUPS in [6]. This means that removing a concept component with arity n can resolve n clashes. To illustrate the benefit of our fine-grained approach, we add the following axioms to Example 1:

$$\alpha_5: B \doteq C \sqcap \forall R.(P \sqcap \forall R.F)$$

$$\alpha_6: P \doteq \forall R.D \sqcap \neg D$$

In this case, concept B is also unsatisfiable due to the existence of a clash in the node of the tree. For simplicity, we do not show the sequences in the clash. We now annotate the concepts, which are involved in the two clashes, with superscript numbers as follows:

$$\alpha_1: A^1 \doteq C^1 \sqcap D^1 \sqcap G \quad \alpha_2: C^{1,2} \doteq E^{1,2} \sqcap F$$

$$\alpha_3: E^{1,2} \doteq \neg D^1 \sqcap (\exists R.D)^2 \quad \alpha_4: H \doteq \forall R.C$$

$$\alpha_5: B^2 \doteq C^2 \sqcap (\forall R^2.(P^2 \sqcap \forall R.F)) \quad \alpha_6: P^2 \doteq \forall R.D \sqcap (\neg D)^2$$

From the above, we can easily see which concepts in the axioms cause which concepts to be unsatisfiable. It is obvious that removing concept E in axiom α_2 can

resolve two clashes.

Removing clashes

Given an unsatisfiable concept A in \mathcal{T} , we can obtain a fully expanded tree containing a node with at least one clash. For each clash, the sequences of the clash, Seq^+ and Seq^- , are obtained by Definition 1. Let the first elements of the sequences be $(a : C, I', -)$ and $(a : -C, I'', -)$, and let the last element of the sequences be $(b : A, \emptyset, nil)$. We know that the set of axioms $I := I' \cup I''$ causes A to be unsatisfiable; we can derive the following lemma:

Lemma 2 *Let \mathcal{D} be the set of all concepts appearing in the elements of the sequences, removing one of the concepts in \mathcal{D} from one of the axioms in I is sufficient to resolve the clash.*

Proof For any concept picked from \mathcal{D} , it must occur in the sequences and have adjacent elements which are before or after. For any two adjacent elements in a sequence, e_1 and e_2 , there are only two possibilities: (1) e_1 and e_2 are in the form of $(a : E_1, -, a : E_2)$ and $(a : E_2, -, -)$ containing the same individual, this means the concept E_1 is a superconcept of E_2 . If E_1 (or E_2) is removed, the subsumption relationship between E_2 and E_1 is removed. Therefore, the individual a does not belong to E_1 (or E_2), nor does it belong to any of the concepts in the elements preceding the occurrence of e_1 in the sequences. Hence the clash is resolved. (2) e_1 and e_2 are in the form of $(a : E_1, -, b : E_2)$ and $(b : E_2, -, -)$ containing different individuals, this means the concept E_1 participates in a role relationship with E_2 . If E_1 or E_2 is removed, then there is no such individual a participating in the role, and all the concepts in the elements preceding the occurrence of e_1 are not related to a , and hence the clash is resolved. ■

Let there be an unsatisfiable concept A in terminology \mathcal{T} . Let's assume the fully expanded tree has n nodes, and there are m clashes in one of the nodes, so there are m pairs of sequences of clashes corresponding to the node. Removing at most m concepts appearing in the elements of all the sequences of the clashes in the node is sufficient to resolve the unsatisfiability. Because a concept C may be involved in t clashes in the node (where $1 \leq t \leq m$), then C appears t times in all the sequences of the clashes in the node, i.e. $arity(C, \alpha) = t$. Removing C in α can resolve t clashes. By Lemma 2 the removal of the elements containing C from all the sequences will remove their relationships with A . Thereafter removing $m - t$ of the concepts appearing in all the sequences is sufficient to resolve the unsatisfiability of A .

3.1 Lost Entailments due to Removal

The simplest way to resolve a clash is to remove the concepts in the problematic axioms or the whole axioms. However, in this case, it will be easy for ontology modellers to accidentally remove the stated or implicit entailments in the ontology. We use a typical bird-penguin example to illustrate the lost entailments: $\alpha_1: Bird^* \sqsubseteq Fly^* \sqcap Animal$; $\alpha_2: Eagle \sqsubseteq Bird$; $\alpha_3: Penguin^* \sqsubseteq Bird^* \sqcap (\neg Fly)^*$ (The concepts tagged with a star are relevant to the unsatisfiability of Penguin). When (parts of) an axiom involved in the unsatisfiability is removed, we calculate the impact of the removal in two ways: (1) other satisfiable named concepts in the ontology may have lost entailments. To resolve the above problem, one may claim that not all birds can fly, therefore, the concept Fly in α_1 is removed. However, the indirect assertion $Eagle \sqsubseteq Fly$ is removed. In general we lose $X \sqsubseteq Y$ where X is a subconcept of Bird and Y is a superconcept of Fly. (2) the unsatisfiable concept may lose entailments which are not relevant to its unsatisfiability. In the above example, if the user removes Bird in α_3 , then the indirectly stated assertion $Penguin \sqsubseteq Animal$ is removed. However, $Penguin \sqsubseteq Animal$ may also be present explicitly in the ontology. Therefore to check if the entailments which are removed (due to the removal of axioms) are really lost entailments, we have to check if the changed ontology still satisfies these entailments. If it does not, then we can say these entailments are lost.

Impact on other satisfiable named concepts

We first describe how to calculate the impact of removal on other satisfiable named concepts in the ontology. Given a terminology \mathcal{T} , we assume a concept E in an axiom $\alpha_i : A_i \doteq C_i$ is to be removed. If E is on the left hand side of the axiom, then that means the whole axiom is to be removed (i.e. the modified axiom will be $A_i \sqsubseteq \top$). If E is at the right hand side of an axiom, then that E in C_i is replaced by \top . Any named concepts in other axioms, which refer to A_i , will lose entailments induced by α_i . The lost entailment of a concept can be computed by the difference of the original and modified concept. To do this we adapt the notion of the “different” operator between concepts which is defined by Teege [8]. The difference of C and C' (1) contains enough information to yield the information in C if added to C' , i.e. it contains all information from C which is missing in C' , and (2) is maximally general, i.e. it does not contain any additional unnecessary information.

Definition 3 (Difference of Concepts) *Let C and C' be the original and modified concept expressions, the*

difference of C and C' , which is a set of concepts, is defined as

$$\text{different}(C, C') = \begin{cases} \max_{\sqsupseteq} \{E \mid E \sqsupseteq C \sqcup \neg C'\} & \text{if } C \sqsubseteq C', \\ \max_{\sqsupseteq} \{E \mid E \sqsupseteq C' \sqcup \neg C\} & \text{if } C' \sqsubseteq C \end{cases}$$

In our example, if concept E in axiom α_2 is removed, then the modified axiom becomes $C \sqsubseteq F$. As α_4 refers to C , the lost entailment of H will be $\forall R.(E \sqcap F) \sqcup \neg \forall R.F$, i.e. $\forall R.E \sqcup \neg \forall R.F$.

Impact on the unsatisfiable concept

Next we describe how to calculate the impact of the removal of (part of) axioms on the unsatisfiable concept. Note that when a concept is unsatisfiable, it is trivially a subconcept of all satisfiable concepts and equivalent to all unsatisfiable concepts. In this case, we need to differentiate between the explicitly stated entailments of the unsatisfiable concepts and the trivial ones. In the following we use an example to explain how to find explicitly stated entailments, which are not relevant to the unsatisfiability, by analysing the sequences of the clash of the unsatisfiable concept.

Our idea is to search for any element which exists in $\mathcal{L}(x)$ but not in the sequences of the clash. In our example, if C in axiom α_1 is going to be removed, then we have to calculate the lost entailments of A which are not relevant to A 's unsatisfiability. Figure 2 shows the node and the sequences of the clash in A . We find that both $(a : A, -, -)$ and $(a : C, \{1\}, a : C \sqcap D \sqcap G)$ exist in Seq^- (cf.1 in Figure 2). We search for elements in $\mathcal{L}(x)$ whose second concept assertion is $a : C$, but which do not exist in Seq^- . $(a : E \sqcap F, -, a : C)$ matches $a : C$ but exists in Seq^- (cf.3), so we keep searching for another element whose second concept assertion is $a : E \sqcap F$. The matched elements are $(a : E, -, a : E \sqcap F)$ (cf.4) and $(a : F, -, a : E \sqcap F)$ (cf.5); $(a : F, -, a : E \sqcap F)$ does not exist in Seq^+ and has the same individual as A . That means F is a superconcept of A , and hence, the lost entailment is $A \sqsubseteq F$. Finding lost entailments with role relationships is quite complicated, and we do not have space to fully explain it here. We give our example: if concept P in axiom α_5 is going to be removed, then the lost entailment is $B \sqsubseteq \forall R.(\forall R.D)$.

3.2 Harmful and Helpful Changes

In this subsection we study ways of changing problematic axioms to resolve the unsatisfiability. It should be noted that improperly rewriting a problematic axiom might not resolve the unsatisfiability, and could introduce additional unsatisfiability. It is important to help ontology modelers to make changes in order to prevent unintended contradictions. For this purpose,

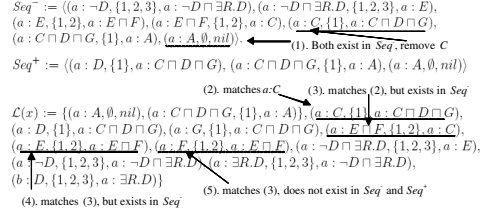


Figure 2. Node and sequences of clash in A

we define *harmful* and *helpful* changes. Deciding if a change is harmful or helpful is therefore a matter of deciding whether the modifications can resolve the clash without invoking additional unsatisfiability, and compensate for the lost entailments due to the change.

Assume a concept E in a problematic axiom α_i : $A_i \sqsupseteq C_i$ is chosen to be replaced by some other concept. We can find the harmful concepts for the replacement of E by analysing the elements in the sequences of clashes in the ontology.

Definition 4 (Harmful Change) A change is harmful with respect to a terminology \mathcal{T} , if some A_i which is satisfiable in \mathcal{T} is not satisfiable in the changed ontology \mathcal{T}' , or there exist some cyclic axioms in \mathcal{T}' . That is, $\mathcal{T} \not\models A_i \sqsupseteq \perp$ and $\mathcal{T}' \models A_i \sqsupseteq \perp$, for some A_i in \mathcal{T} , or \mathcal{T}' is not unfoldable.

Lemma 5 Assume a concept C in axiom α_i is to be rewritten. Let Seq^+ and Seq^- be two sequences of a clash involving C . If one of the elements, e , in Seq^+ is of the form $(a : C, I, -)$ and $i \in I$, then the following concepts are harmful for replacing C : (1) all the concepts in the elements in Seq^+ , which contain the same individual as e ; (2) the negation of all the concepts in elements in Seq^- , which contain the same individual as e . (The lemma is analogous for the element e existing in Seq^- .)

Proof Assume that a concept C in axiom α_i is to be rewritten, and two sequences of a clash involving C , Seq^+ and Seq^- , are obtained from a node x of the tree \mathbf{T} . In a sequence, for every two adjacent elements, e_1 and e_2 , which are of the form $(a : E_1, -, a : E_2)$ and $(a : E_2, -, -)$, containing the same individual, the concept E_1 is a superconcept of E_2 . This extends inductively to all elements preceding e_1 , i.e. they are all superconcepts of E_2 .

(1) If an element e in Seq^+ , which is of the form $(a : C, I, -)$ and $i \in I$, then the concepts in all the elements, which are preceding e and contain individual a , are harmful for replacing C , because they are

superconcepts which are involved in the clash. Those succeeding e and containing the individual a , which are subconcepts of C , cannot be used for replacement because this will introduce cyclic axioms.

(2) The elements in Seq^- are leading to a negated concept, which results in a contradiction. Hence, the negation of all the concepts in elements in Seq^- , which contain the same individual a , are also harmful. ■

In our example, if C in axiom α_1 is going to be replaced, we know that there exists an element $(a : C, \{1\}, a : C \sqcap D \sqcap G)$ in Seq^+ of the clash, then the harmful concepts for the replacements will be $\neg D$, $\neg D \sqcap \exists R.D$, E , $E \sqcap F$, $C \sqcap D \sqcap G$, A , $\neg D$, $\neg A$, and $\neg(C \sqcap D \sqcap G)$.

If we know which concepts are harmful to replace a concept in a problematic axiom, then all the concepts, which are not harmful, can be candidates for the replacement. However, there are many possible candidates of such concepts. Our aim is to find desirable concepts for replacement in order to minimise the impact of changes. To do this we introduce *helpful* changes which cover for the lost entailments due to the removal. When an axiom $A \doteq C$ in \mathcal{T} is changed to be $A \doteq C'$, this change is helpful if (1) C' can compensate for at least one lost entailment due to the removal of C , (2) the changes are not harmful, that means all concepts which are satisfiable in \mathcal{T} are also satisfiable in the changed ontology, and \mathcal{T} remains in an unfoldable format. Note that we only rewrite the concepts in the right hand side of axioms. We now formally define a helpful change.

Definition 6 (Helpful Change) Assume that a problematic axiom α in \mathcal{T} is going to be changed, \mathcal{T}' is the changed ontology. The change is helpful with respect to \mathcal{T} , if the following conditions hold:

- if Ω is the set of lost entailments in \mathcal{T} due to the removal of α , such that $\forall \gamma \in \Omega, \mathcal{T} \models \gamma$, let the intermediate ontology be $\mathcal{T}_1 = \mathcal{T} \setminus \Omega$, then there exists $\beta \in \mathcal{T}'$, such that $\mathcal{T}_1 \not\models \beta$ and $\mathcal{T}' \models \beta$;
- for all $A_i \doteq C_i \in \mathcal{T}$, $\mathcal{T} \not\models A_i \doteq \perp$ and $\mathcal{T}' \not\models A_i \doteq \perp$; \mathcal{T}' is unfoldable.

Lemma 7 Assume C in a problematic axiom is going to be replaced by C' , the change is helpful if C' is a superconcept of C and is not involved in the clash.

Proof It is obvious that any concept which is not involved in the clash is not harmful as a replacement for C . We now prove its superconcepts are helpful. Given that in an axiom $\alpha : A \doteq C$ in \mathcal{T} , concept C is going to be replaced by its superconcept C' . We divide the

change into two steps:

- (1) remove C from α , the lost entailment is $\{A \doteq C\}$, the changed ontology $\mathcal{T}_1 = \mathcal{T} \setminus \{A \doteq C\}$;
- (2) add C' to α , the final ontology $\mathcal{T}' = \mathcal{T}_1 \cup \{A \doteq C'\}$. As C' is a superconcept of C , $A \doteq C$ is removed in \mathcal{T}_1 , so the indirect subsumption relationships of A with C 's superconcepts are also removed, that means $\mathcal{T}_1 \not\models A \sqsubseteq C'$, but obviously, $\mathcal{T}' \models A \sqsubseteq C'$. ■

Lemma 8 Given two sequences of a clash obtained from the set $\mathcal{L}(x)$, assume a concept C in axiom α_i is to be rewritten. C' is helpful as a replacement for C , if the following conditions are held: (1) there exist two elements e and e' in $\mathcal{L}(x)$, which are $(a : C, I, -)$ and $(a : C', I', -)$, and no element of the form $(a : C', I', -)$ exists in either of the two sequences; (2) $I \subset I'$, the index-set of the element with concept C is a proper subset of that of the element with concept C' .

Proof (1) If an element $(a : C', -, -)$ exists in $\mathcal{L}(x)$, but not in either of the two sequences, then C' is not involved in the clash. (2) If elements e and e' in $\mathcal{L}(x)$ contain the same individual, then C is either a subconcept or superconcept of C' . Additionally, if the index-set of the element e is a proper subset of that of the element e' , then that means e' is added to $\mathcal{L}(x)$ after the addition of e , thus, C' is a superconcept of C . ■

Continuing our example, if C in axiom α_1 is going to be replaced, and there exists an element $(a : C, \{1\}, a : C \sqcap D \sqcap G)$ in Seq^+ of the clash, then we know that the two elements $(a : G, \{1\}, a : C \sqcap D \sqcap G)$ and $(a : F, \{1, 2\}, a : E \sqcap F)$ are not involved in the sequences of the clash. However, the former element does not fulfill condition (2) in Lemma 8, because the index set of $a : C$ is not a proper subset of $a : G$. Hence, the only helpful concept for the replacement is the concept of the latter element, F .

4 Preliminary Evaluation

To demonstrate the use of our proposed approach, we have built a prototype which we aim to eventually extend to an interactive ontology management tool for debugging ontologies. Benchmarking with real-life ontologies is obviously a convincing way to evaluate the quality of our algorithms. However, there is only a limited amount of realistic ontologies that are both represented in \mathcal{ALC} and unsatisfiable. We therefore constructed simplified \mathcal{ALC} versions for two ontologies. Without losing the unsatisfiability, we removed numerical constraints, role hierarchies, instance information, etc., and ensured they were transformed to an unfoldable format. The algorithms described above are implemented in Java. The tests were performed on a PC⁴

⁴Intel Pentium IV with 2.4GHz and 1GB RAM

with Windows XP SP2 as operating system. Figure 3 (A) shows the screenshot of our system in which the Mad_Cow ontology⁵ was imported. It contains 54 concepts; mad_cow was detected with a clash. The run time was 1.974 sec. The axioms which cause mad_cow to be unsatisfiable are listed (cf. (B)); the parts of axioms relevant to the unsatisfiability are highlighted (cf. (C)). If the user removes the axiom “cow \sqsubseteq vegetarian”, the lost entailments of this removal can be previewed. The harmful and helpful changes are also listed.

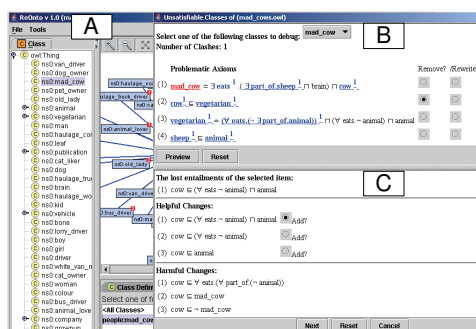


Figure 3. Screenshot of implemented tool

The previous paragraph presents the results of the approach on debugging an ontology that was specifically designed to demonstrate our fine-grained approach. The outcomes illustrated that the parts of the axioms which cause unsatisfiability were successfully identified. We now present results from the implementation of a real-world ontology, which is the Earth Realm⁶ ontology. As it is a satisfiable ontology, we randomly changed it such that each change on its own lead to unsatisfiable concepts. For example, we added disjointness statements among sibling concepts (as suggested in [5]). Similarly the ontology was simplified to an unfoldable \mathcal{ALC} format. Of the 236 concepts, 28 were unsatisfiable. The run time was 27.852 sec. As it is well known that the algorithm for unfoldable \mathcal{ALC} TBoxes is computationally difficult; the running time for finding the clashes of concepts increases rapidly with increasing number of concepts in the ontology.

5 Related Work

To our knowledge, the most relevant works are described in [6, 3]. Schlobach et al. [6] apply syntactic generalisation techniques to highlight the exact position of a contradiction within the axioms of the TBox.

⁵<http://cohse.semanticweb.org/ontologies/people.owl>

⁶<http://sweet.jpl.nasa.gov/ontology/earthrealm.owl>

This is called ‘concept pinpointing’. However they do not support users in changing axioms and minimising the impact of changes. Kalyanpur et al. [3] extend the tableaux tracing algorithm to indicate specific parts of axioms that are responsible for the unsatisfiability. They suggest rewriting the axiom based on common errors in OWL ontology modelling. However, the common error patterns may only apply for those ontologies built by non-expert users, it is insufficient to cover other applications, such as ontology integration. Moreover, by correcting ontologies based on the common error patterns, one may arrive at an unsatisfiable ontology.

6 Conclusion

In this paper we have proposed a fine-grained approach to rewriting the problematic axioms in an ontology, by revising the classical tableaux algorithm. Our algorithm not only identifies the problematic axioms, but also captures which part(s) of the axioms are responsible for the unsatisfiability of concepts. Moreover, we present the methods of finding harmful and harmless changes for concepts which are going to be replaced. Our preliminary evaluation implemented by an interactive debugging tool has demonstrated the applicability of our approach in practice. Such a tool could be very helpful for users who want to diagnose problematic axioms on a fine-grained level and achieve desired satisfiable ontologies.

References

- [1] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. *J. Autom. Reasoning*, 14(1):149–180, 1995.
- [2] F. Baader and W. Nutt. Basic description logics. In *The Description Logic Handbook: Theory, Implementation, and Applications*.
- [3] A. Kalyanpur, B. Parsia, E. Sirin, and B. Cuenca-Grau. Repairing Unsatisfiable Concepts in OWL Ontologies. In *ESWC-2006*, June 2006.
- [4] T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding maximally satisfiable terminologies for the description logic ALC. In *Proceedings of AAAI-06*, July 2006.
- [5] S. Schlobach. Debugging and semantic clarification by pinpointing. In *Proceedings of ESWC*, 2005.
- [6] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI’03*. Morgan Kaufmann, 2003.
- [7] M. Schmidt-Schaub β and G. Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, 1991.
- [8] G. Teege. Making the difference: A subtraction operation for description logics. In *the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR94)*, 1994.