Debugging Overgeneralized Concepts in $\mathcal{ALCN}$ Terminologies

Ekaterina Ovchinnikova and Kai-Uwe Kühnberger

University of Osnabrück
e.ovchinnikova@gmail.com, kkuehnbe@uos.de

Abstract. Unsatisfiability of terminological knowledge and generalization mistakes are crucial problems in the domain of ontology construction and extension. Existing approaches to ontology debugging remove problematic axioms or rewrite them. Whereas removal or neglect of axioms imply a loss of knowledge, the existing axiom rewriting techniques handle relatively weak logics and additionally cause an undesirable loss of entailments. We present an approach that repairs overgeneralization in terminologies formalized in $\mathcal{ALCN}$ description logic. Our approach is knowledge-preserving, i.e. as few entailments are lost as possible.

1 Introduction

Ontological knowledge has been shown to be the backbone of the Semantic Web. It is used in many distributed applications requiring a common semantic background on which they exchange data, resources, and services. In order to make this information usable for automatic reasoning procedures, it has to be formalized within a logical framework such as Description Logics (DL) [Ba1]. In the past years, a variety of successful systems have been developed that make use of markup standards based on DL with varying degrees of expressiveness (see [Ba1] for an overview of different versions of description logic).

The OWL[1] standard has become an essential format of exchange for many semantic web applications, but also for reasoning systems such as Racer[2], FaCT[3], or KAON2[4]. Developing an ontology that respects OWL has therefore the advantage that an important set of processing tools is immediately available. However, storage of ontological information within a logical framework entails inconsistency problems, because information can contradict each other, making the given ontology unsatisfiable and therefore useless for reasoning purposes. The problem of inconsistency becomes even more important with regard to large-scale ontologies: resolving inconsistencies in large ontologies by hand is time-consuming and tedious, therefore automatic procedures are required to debug ontologies.

[1] http://www.w3.org/TR/owl-features/
[2] http://www.sts.tu-harburg.de/ r.f.moeller/racer/
[3] http://www.cs.man.ac.uk/ horrocks/FaCT/
[4] http://kaon2.semanticweb.org/

Standard DL reasoning systems help to find unsatisfiable concepts, but do not support users in resolving unsatisfiability. There is a number of approaches to detecting sets of axioms that are responsible for unsatisfiability (cf. [SC1,Wa1]). These approaches do not give a solution to the question which parts of the problematic axioms should be debugged and how. Then, some approaches suggest to remove problematic axioms (cf. [Ha1,Fa1]). Finally, there are approaches that propose to rewriting problematic parts of axioms (cf. [Ka1,La1,OK1]). As shown in section 4, the existing approaches cause undesired loss of entailments in the ontology and consider only relatively unexpressive DLs.

The approach presented in this paper focuses on unsatisfiability in $\mathcal{ALCN}$ terminologies and especially on overgeneralized concepts. Overgeneralization can occur, if the definitions of some concepts in the ontology are too general so that they cause a contradiction with unforeseen exceptions. We propose a regeneralization procedure that rewrites problematic concept definitions in order to achieve a consistent terminology. The introduced procedure tries to remove as little information as possible from the ontology. The given approach represents an integration of the ideas first expressed in [La1] and [OK1] and extends the underlying description logic to $\mathcal{ALCN}$.

This paper is organized as follows: We first present the formal background of the underlying logics (section 2) and describe the sources of unsatisfiability of a DL terminology (section 3). In section 4, we give a short overview of the main approaches to automatic ontology debugging. Section 5 presents our approach and the corresponding implementation algorithm. In the last two sections, we discuss open questions and problems and conclude the paper.

2 $\mathcal{ALCN}$ Description Logic

Description logics (DL) are well-known as a widely developed model-theoretic formalism designed for knowledge representation and reasoning (see [Ba1] for an overview). In present paper, we consider $\mathcal{ALCN}$ terminologies. Let N_C and N_R be sets of *concept names* and *role names* respectively. A DL-terminology consists of *terminological axioms* (*TBox*) of the form $C \sqsubseteq D$ or $C \equiv D$ where C and D are concept descriptions specified in $\mathcal{ALCN}$-DL ($A \in N_C, R \in N_R, n \in \mathbb{N}$) as follows:

$$C \rightarrow \top \mid \bot \mid A \mid \neg C \mid \forall R.C \mid \exists R.C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \leq nR \mid \geq nR$$

The symbol $\doteq$ denotes the syntactical equality of concept descriptions. The semantics of concepts and axioms is defined in the usual model-theoretic way (see [Ba1]) in terms of an *interpretation function* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set of individuals and the function $\cdot^{\mathcal{I}}$ maps every concept name A to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every role name R to $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. $\mathcal{I}$ is called a *model* of a TBox $\mathcal{T}$ iff for every $C \sqsubseteq D \in \mathcal{T}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and for every $C \equiv D \in \mathcal{T}$: $C^{\mathcal{I}} = D^{\mathcal{I}}$. A concept description D *subsumes* C towards $\mathcal{T}$ ($\mathcal{T} \models C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. A concept description C is called *satisfiable* in $\mathcal{T}$ iff there is a model M of $\mathcal{T}$ such that $C^M \neq \varnothing$. A terminology is unsatisfiable if it contains an unsatisfiable atomic concept.

If $\langle x, y\rangle \in R^I$ then y is an *R-successor* of x. A TBox $\mathcal{T}$ is *unfoldable* if it is acyclic and every axiom in $\mathcal{T}$ contains only an atomic concept on the left side.

3 Sources of Unsatisfiability in DL Terminologies

A DL ontology can be unsatisfiable only if its underlying logic allows negation. Ontologies share this property with every logical system (like, for example, first-order logic). For the approaches concerned with core ontologies no contradictions in the ontological knowledge base is possible. But for approaches using more powerful logics, the problem of inconsistency becomes very important [Ha1].

In practice, logical conflicts can be caused by several reasons. For example, errors in the automatic ontology learning procedure or mistakes of the ontology engineer can generate unintended contradictions. Another type of logical contradiction is connected with polysemy. If an ontology is learned automatically, then it is hardly possible to distinguish between word senses. Suppose, the concept *tree* is declared to be a subconcept both of *plant* and of *data structure* (where *plant* and *data structure* are disjoint concepts). Both of these two interpretations of *tree* are correct, but it is still necessary to describe in the ontology two different concepts with different identifiers (e.g. *TreePlant*, *TreeStructure*). Otherwise, the terminology remains unsatisfiable.

Finally, there is a set of problems connected with generalization mistakes.

Example 1 *1.* $Bird \sqsubseteq CanFly$ *2.* $CanFly \sqsubseteq CanMove$ *3.* $Canary \sqsubseteq Bird$ *4.* $Penguin \sqsubseteq Bird \sqcap \neg CanFly$

In Example 1, the statement *birds can fly* is too general. If an exception occurs (*penguin*), the ontology becomes unsatisfiable, since penguin is declared to be a bird, but it can not fly.

In this paper, we are mainly concerned with the overgeneralization problem. The general questions we consider are: (1) Having a contradiction in a terminology, how can we detect axioms that are responsible for this contradiction? (2) How can we define which axiom should be modified to resolve the contradiction? (3) How can we modify the terminology in order to resolve the contradiction so that as much entailments as possible are retained.

4 Related Work

In the past few years, a number of new approaches to automatic ontology debugging were suggested. A technique to find a minimal set of axioms that are responsible for the inconsistency in an ontology was first proposed in [BH1]. To detect a set of problematic axioms this approach labels assertions and traces these labels back after finding a contradiction in an tableau expansion tree.

[SC1] have advanced this idea and have applied it to debugging unsatisfiable terminologies. The notions *minimal unsatisfiability-preserving sub-TBoxes*

(MUPS) and *minimal incoherence-preserving sub-TBoxes* (MIPS) were introduced and formalized. [SC1] propose also an axiom pinpointing service for $\mathcal{ALC}$-DL identifying the exact parts of axioms that are responsible for a contradiction.

Several present approaches to ontology debugging are concerned with explanation services integrated into ontology developing tools. For example, [Wa1] present a service explaining unsatisfiability in OWL-DL ontologies by highlighting problematic axioms and giving natural language explanations of the conflict.

In [Ha1], an approach to automatic ontology extraction is described. Every extracted axiom receives a confidence rating showing how frequent the given axiom occurs in external sources. If the resulting ontology is inconsistent, the set of problematic axioms is detected and the axiom with the lowest confidence rating is removed.

Unfortunately, the approaches sketched above either do not give solutions of how to fix the discovered contradictions or simply propose to remove a problematic part of an axiom where this axiom is chosen according to some rating (see section 5). Removing parts of axioms can entail an undesirable loss of information. Consider Example 1: If the concept $CanFly$ will be removed from axiom 1 (this solution seems to be natural), then the entailments $Bird \sqsubseteq CanMove$ and $Canary \sqsubseteq CanFly$ will be lost.

In [Fa1], inductive logic programming techniques are proposed to resolve ontological inconsistencies. The authors present an ontology refinement procedure based on positive and negative assertions for concepts. If a concept C is unsatisfiable after an ontology extension, then the axiom defining C is replaced by a new axiom, constructed on the basis of the positive assertions for this concept. Therefore, the information previously defined in the ontology for C gets lost.

In [Ka1], the tableau algorithm for OWL-DL is extended with a tracing technique in order to detect conflicting parts of axioms. [Ka1] suggests to rewrite axioms using frequent error patterns occurring in ontology modeling. However the application of the error patterns does not guarantee the satisfiability of the changed terminology.

[La1] revise the axiom tracing technique proposed in [BH1] and support ontology engineers in rewriting problematic axioms. In [La1], not only conflicting parts of axioms are detected, but also a rating technique for calculating which part of which axiom should be modified is proposed. Moreover, a concept that should replace the problematic part of the chosen axiom is constructed. This approach is realized for $\mathcal{ALC}$-DL. Returning to Example 1, this approach will keep the entailment $Bird \sqsubseteq CanMove$, but not the entailment $Canary \sqsubseteq CanFly$.

An approach to regeneralization of overgeneralized concepts conflicting with exceptions was presented in [OK1]. This approach implies not only rewriting a problematic axiom, but also splitting an overgeneralized concept C into a more general concept that do not conflict with exceptions and a more specific one that captures the original semantics of C and is declared to subsume former subconcepts of C except the exceptions. This approach will also keep the entailment $Canary \sqsubseteq CanFly$. In [OK1], a regeneralization algorithm based on structural subsumption for the unexpressive $\mathcal{ALE}$-DL is developed.

5 Proposed Approach

5.1 Tracing Clashes in $\mathcal{ALCN}$ Model Trees

In this section, we revise the tableau-based algorithm presented in [La1] for tracing clashes in unsatisfiable $\mathcal{ALC}$-terminologies and rewriting problematic axioms. We extend the description logic under consideration to $\mathcal{ALCN}$ by adding number restrictions and consider the entailments that are lost after rewriting.

Following [La1], suppose that a terminology $\mathcal{T}$ contains axioms $\{\alpha_1, ..., \alpha_n\}$, where α_i refers to the axiom $A_i \sqsubseteq C_i$ or $A_i \equiv C_i$ $(i \in \{1, ..., n\})$. Checking the satisfiability of a concept description C the tableau algorithm constructs a model of C represented by a tree $\mathbf{T}$. Each node x in this tree is labeled with a set $\mathcal{L}(x)$ containing elements of the form $(a : C, I, a' : C')$, where C and C' are concept descriptions, a and a' are individual names and I is a set of axiom indices. An element $(a : C, I, a' : C')$ means that the individual a belongs to the concepts description C due to the application of an expansion rule on C' and I contains the indices of the axioms where $a : C$ comes from.

$\mathbf{T}$ is initialized with a node x and $\mathcal{L}(x) = \{(a : C, \varnothing, nil)\}$. The algorithm expands $\mathbf{T}$ according to the rules in Table 1. Concept descriptions involved in the expansion are converted to negation normal form. The algorithm terminates if no more expansion rules can be applied to tree nodes.

$\mathbf{T}$ contains a *clash* if an individual a belongs simultaneously to concept descriptions C and $\neg C$ or $\leq nR$ and $\geq mR$ with $n < m$. Every time when a new element $e = (a : C, -, -)$[5] is added to $\mathcal{L}(x)$ the algorithm checks if e introduces a new clash where a is involved. If this is the case, the element (x, E) (where E contains elements from $\mathcal{L}(x)$ that constitute the clash) is added to the set $Clashes$[6].

A clash in an expansion tree does not necessarily mean unsatisfiability. A concept description containing disjunction is unsatisfiable only if every branch of the expansion tree contains a clash. But our algorithm searches for all clashes in every single branch, because an unsatisfiable alternative in a concept definition contains no information and, thus, does not make much sense in the ontology. The same story is true for value restrictions. If an individual a in a model tree for a concept A belongs to a value restriction $\forall R.C$, where C is unsatisfiable, but a has no R-successors, then this value restriction does not cause unsatisfiability of A. But with the advent of instances or subconcepts of A which have R-successors this value restriction will invoke unsatisfiability, therefore our algorithm fixes it.

The situation with at-most number restrictions is more complicated. If an individual belongs to more then n existential restrictions and to a number restriction $\leq nR$, then it is necessary to merge the existential restrictions into n restrictions. In most of the cases there are many merging combinations and unsatisfiability arises only if every of these combinations contains a clash. Reparing all merging combinations is senselessly (in contrast to the situation with disjunction) and costly. A better solution is to choose one combination and to resolve

[5] "-" is a place holder. It stands for any value.

[6] Hereinafter we use the term *clash* to refer to the elements of the set $Clashes$.

clashes in it. The set $EquivTrees$ collects equivalence classes of model trees that are united according to an at-most number restriction that is responsible for their formation.

The Rule $\leq$ in Table 1 requires special comments. Having found a node a that belongs to a number restriction $\leq nR$ and has more than n R-successors, the algorithm generates all possible merging combinations of these R-successors into n successors. If a newly generated tree contains a clash $c = (x, E)$, then the algorithm checks whether this clash traces to the merged R-successors. If this is true, then the given number restriction is added to E as a source of the clash c.

Definition 1 *(Trace)*
Given an element $e = (a_0 : C_0, I_0, a_1 : C_1)$ *in a tree* $\mathcal{L}(x)$*, the trace of* e *is a sequence of the form* $\langle (a_0 : C_0, I_0, a_1 : C_1), (a_1 : C_1, I_1, a_2 : C_2), \ldots, (a_{n-1} : C_{n-1}, I_{n-1}, a_n : C_n), (a_n : C_n, \varnothing, nil)\rangle$*, where* $I_{i-1} \subseteq I_i$ *for each* $i \in \{1, ..., n\}$ *and every element in the sequence belongs to* $\mathcal{L}(x)$*.*

The expansion algorithm returns a number of model trees, a set of clashes and a set of equivalence classes $EquivTrees$ such that only one tree from every class has to be repaired. Every expanded tree $\mathcal{L}(x)$ that does not occur in $EquivTrees$ is added to the set $EquivTrees$ as (x, nil).

The given algorithm extends the canonical satisfiability tableau algorithm for $\mathcal{ALCN}$ [BS1] only with checking additionally for every at-most number restriction whether this restriction introduces a clash. Thus, the properties of the original algorithm that can be found in [BS1] are also relevant for our algorithm.

5.2 Resolving Clashes

Given a clash (x, E) obtained from a model tree $\mathcal{L}(x)$ for a concept A, the union of the index sets of the elements in E gives the set of axioms which causes the clash. Concerning unsatisfiability of A this set of axioms corresponds to MUPS. [La1] shows that removing one of the concepts appearing in the clash traces from the appropriate parts of axioms from I is sufficient to resolve the clash where the conflict is of the form $a : C$ and $a : \neg C$. It is easy to show that this claim is also true for the clashes of the form $a :\leq nR$ and $a :\geq mR$, $n < m$. Other things being equal, removal[7] of an atomic concept or of a number restriction causes less lost entailments than removal of a relational restriction or a complex concept description containing conjunction or disjunction.

One of the most important pragmatic problems is selecting which of the problematic axioms needs to be removed or modified. A lot of ranking criteria were suggested in the literature on ontology debugging (cf. [SC1,Ka1,La1,Ha1]):

- *Arity* of an axiom α shows in how many clashes α is involved. The higher is the arity, the lower is the rank of α.
- *Semantic impact* of α shows how many entailments are lost if α is removed. Axioms with a hight semantic impact are ranked higher.

[7] Under removal of C we mean the replacement of C with $\top$.

Table 1. Tableau expansion rules for $\mathcal{ALCN}$

Rule $\equiv+$	if $A_i \equiv C_i \in \mathcal{T}$, and $(a : A_i, I, a' : A')$ is not tagged, then $tag((a : A_i, I, a' : A'))$ and $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : C_i, I \cup \{i\}, a : A_i)\}$
Rule $\equiv-$	if $A_i \equiv C_i \in \mathcal{T}$, and $(a : \neg A_i, I, a' : A')$ is not tagged, then $tag((a : \neg A_i, I, a' : A'))$ and $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : \neg C_i, I \cup \{i\}, a : \neg A_i)\}$
Rule $\sqsubseteq$	if $A_i \sqsubseteq C_i \in \mathcal{T}$, and $(a : A_i, I, a' : A')$ is not tagged, then $tag((a : A_i, I, a' : A'))$ and $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : C_i, I \cup \{i\}, a : A_i)\}$
Rule $\sqcap$	if $(a : C_1 \sqcap C_2, I, a' : A') \in \mathcal{L}(x)$, then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : C_1, I, a : C_1 \sqcap C_2)\} \cup \{(a : C_2, I, a : C_1 \sqcap C_2)\}$
Rule $\sqcup$	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 $\mathcal{L}(z) := \mathcal{L}(x) \cup \{(a : C_2, I, a : C_1 \sqcup C_2)\}$
Rule $\exists$	if $(a : \exists R.C, I, a' : A') \in \mathcal{L}(x)$, and the above rules cannot be applied, $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(b : C, I, a : \exists R.C)\}$, where b is a new individual name not occurring in $\mathcal{L}(x)$
Rule $\leq$	if $(a : \ \leq nR, I, a' : A') \in \mathcal{L}(x)$, and the above rules cannot be applied, then if for $S = \{(b : C, J, a : \exists R.C) \in \mathcal{L}(x)\}$ $\ \|S\| > n$, then for every n-tuple $< S_1, ..., S_n >$ such that $\forall\ 1 \leq i, j \leq n$: $e \in S_i \rightarrow (e \in S$ and $e \notin S_j)$ $\mathcal{L}(y) := \mathcal{L}(x)$ and for every $i \in \{1, ..., n\}$: if $(b : C, J, a : \exists R.C) \in S_i$, then replace b with b_i in $\mathcal{L}(y)$, where b_i is a new individual name not occurring in $\mathcal{L}(y)$ end for Add clashes from $\mathcal{L}(y)$ to $Clashes$ for every $(y, E) \in Clashes$ such that $\exists e_1, e_2 \in E$: $\exists S_{i \in \{1,...,n\}}$: $(b_1 : B, J_1, X_1), (b_2 : C, J_2, X_2) \in S_i$ and $(b_i : B, J_1, X_1) \in Trace_{\mathcal{L}(y)}(e_1)$ and $(b_i : C, J_2, X_2) \in Trace_{\mathcal{L}(y)}(e_2)$ Add $(a : \ \leq nR, I, a' : A')$ to E end for Add $(y, (a : \ \leq nR, I, a' : A'))$ to $EquivTrees$ end for
Rule $\forall$	if $(a : \forall R.C, I, a' : A') \in \mathcal{L}(x)$, and the above rules cannot be applied, then for every $(b : D, J, a : \exists R.D) \in \mathcal{L}(x)$: $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(b : C, I, a : \forall R.C)\}$ and if there is no $(b : D, J, a : \exists R.D) \in \mathcal{L}(x)$, then if there is $(b : D, J, a : \forall R.D) \in \mathcal{L}(x)$, then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{b : C, I, a : \forall R.C\}$ else $\mathcal{L}(x) := \mathcal{L}(x) \cup \{b : C, I, a : \forall R.C\}$, where b is a new individual name

– *Syntactic relevance* shows how often concept and role names used in an axiom α are used in other axioms in the ontology. Axioms containing elements that are frequent in the ontology are ranked higher.
– *Manual ranking* of α is provided by the ontology engineer.
– *Frequency ranking* of α is used in approaches to semi-automatic ontology extraction and shows how often concepts and roles in α occur in the external data source.

In this paper, we do not discuss ranking strategies and suppose that one of these strategies has been applied and the problematic axiom to be rewritten has been chosen. Assume a concept description C is chosen to be removed from an axiom α in order to resolve a clash in a model tree for a concept A. The next questions is: How can we find an appropriate concept description C' that can resolve the clash by replacing C? Suppose $\mathcal{T}'$ is obtained from $\mathcal{T}$ by replacing C with C' in α and $\mathcal{T}''$ is obtained from $\mathcal{T}$ by replacing C with $\top$. [La1] suggests C' to be helpful for the replacement if a) C' restores some entailments that are implied by $\mathcal{T}$ but not by $\mathcal{T}''$ and b) there is no atomic concept that is satisfiable towards $\mathcal{T}''$ and unsatisfiable towards $\mathcal{T}'$. Unfortunately, this definition seems to be insufficient. An assumption that a replacement guarantees the elimination of a clash for A if this replacement does not cause concepts that are satisfiable towards $\mathcal{T}''$ to be unsatisfiable towards $\mathcal{T}'$ implies the fact that A is satisfiable towards $\mathcal{T}''$. But if model trees for A contain more than one clash then A can remain unsatisfiable towards $\mathcal{T}''$ even if the removal of C has resolved one clash.

Example 2 *1.* $A \sqsubseteq C \sqcap G \sqcap \exists R.C \sqcap \forall R.G$ *2.* $C \sqsubseteq B \sqcap D$ *3.* $B \sqsubseteq \neg G$

In Example 2, the concept A is unsatisfiable, but the removal of the clash $A \sqsubseteq G, A \sqsubseteq \neg G$ by the replacement of C with $\top$ does not make A satisfiable, because the clash $A \sqsubseteq \exists R.\neg G, A \sqsubseteq \forall R.G$ remains.

Thus, we seek for a replacement that resolves the clash, does not cause new clashes or entailments, and preserves as much entailments implied by $\mathcal{T}$ as possible. A good candidate for such a replacement is a subsumer of C towards $\mathcal{T}$, because a non-subsumer will necessarily introduce new entailments that are not implied by $\mathcal{T}$.

Note that if a concept C is removed from an axiom $A \sqsubseteq C \sqcap B$, then not only the entailment $A \sqsubseteq C$ for A can be lost.

Example 3 *1.* $A \sqsubseteq \exists R.C \sqcap \exists R.\neg C \sqcap \forall R.G$ *2.* $G \sqsubseteq \neg C$

Suppose the concept C in Example 3 is chosen to be removed from $\exists R.C$ in axiom 1, because it conflicts with the value restriction $\forall R.G$. Replacing $\exists R.C$ with $\exists R.\top$ we lose the entailment $A \sqsubseteq \geq 2R$ which is implied by $\exists R.C \sqcap \exists R.\neg C$. But if the entailment $A \sqsubseteq \exists R.C$ is considered to be "wrong" and conflicting, then we probably do not wand to keep the information induced from $\exists R.C$. Therefore we consider a non-conflicting subsumer of a problematic concept C to be a good substitute for C and do not keep the lost entailments implied by C.

Definition 2 *(Minimal nonconflicting subsumer)*
Assume the following is given: a terminology $\mathcal{T}$, a clash (x, E) obtained from the set $\mathcal{L}(x)$, a concept description C that is chosen to be removed from an axiom α_i ($\alpha_i \in \mathcal{T}$ and $\exists e \in E : (a : C, \{i, -\}, -) \in Trace(e)$), an axiom α' obtained from α_i by replacing C with C' and $\mathcal{T}' := \mathcal{T} \setminus \{\alpha_i\} \cup \{\alpha'\}$, and an axiom α'' obtained from α_i by replacing C with $\top$ and $\mathcal{T}'' := \mathcal{T} \setminus \{\alpha_i\} \cup \{\alpha''\}$. A minimal nonconflicting subsumer (MNS) of C is a concept description C' with the following properties:
1) C' subsumes C towards $\mathcal{T}$.
2) The set $Clashes_{\mathcal{T}'}$ contains the same number of clashes as $Clashes_{\mathcal{T}''}$.
3) There exists no entailment β such that $\mathcal{T}'' \not\models \beta$ and $\mathcal{T}' \models \beta$.
4) There exists no concept description C'' with the same properties such that C' subsumes C'' towards $\mathcal{T}$.

Due to condition (1) in Definition 2 MNS introduces no entailments that are not implied by $\mathcal{T}$. Condition (2) guarantees that MNS resolves the clash e and does not introduce new clashes. Condition (3) excludes entailments that are not connected with the clash but are implied by C (see Example 3). Condition (4) stating the minimality of MNS guarantees that there is no other nonconflicting replacement for C that preserves more entailments than MNS.

Lemma 1. *Given a concept description C that is chosen to be removed from an axiom α_i there exists exactly one MNS of C.*

Proof. $\top$ always satisfies the conditions (1) – (3) of Definition 2, therefore if there is no other candidate, then $\top$ is MNS of C. Suppose C_1 and C_2 are MNS of C. Then $C_1 \sqcap C_2$ also satisfies conditions (1)–(3) of Definition 2 and $C_1 \sqcap C_2 \sqsubseteq C_1$ and $C_1 \sqcap C_2 \sqsubseteq C_2$. Thus, C_1 and C_2 are not minimal towards subsumption and, according to the condition (4) of Definition 2, they can not be MNS of C. q.e.d.

According to Definition 2 concept D from Example 2 is an MNS for C. Lemma 2 shows how to calculate MNS. If a concept description chosen to be removed is a number restriction $\leq nR$ that causes impossible mergings of existential restrictions, then the existential restrictions imply an $\geq mR$ number restriction with $n < m$. In order to find MNS for $\leq nR$ we need to induce m from the given set of existential restrictions. As described in [Br1], to induce an at-least number restriction from $\exists R.C_1 \sqcap ... \sqcap \exists R.C_k$ one needs to find out which $C_{i \in \{1,...,k\}}$ are disjoint. Let us denote an at-least number restriction on R induced from a concept description C as $min_R(C)$. $min_R(C)$ can be computed in polynomial time with an oracle for subsumption (see [Br1]).

Lemma 2. *Given a clash (x, E) obtained from the set $\mathcal{L}(x)$ and a concept description C that is chosen to be removed from an axiom α_i ($\alpha_i \in \mathcal{T}$ and $\exists e \in E : (a : C, \{i, -\}, -) \in Trace(e)$), a concept description C' is MNS of C iff the following conditions hold:*

1. *If $C \doteq \geq mR$ and $(a : \leq nR, -, -) \in E$ for $n < m$, then $C' \doteq \geq nR$.*

2. *Else if* $C \doteq \leq nR$ *and* $(a : \geq mR, -, -) \in E$ *for* $n < m$, *then* $C' \doteq \leq mR$.
3. *Else if* $C \doteq \leq nR$ *and there is no* $(a : \geq mR, -, -) \in E$ *for* $n < m$, *then* $C' \doteq \leq kR$, *where* $k = min_R(\bigsqcap_{(b:D,I,a:\exists R.D)\in\mathcal{L}(x)} \exists R.D)$.
4. *Else* $C' \doteq C_1 \sqcap ... \sqcap C_n$, *where for every* $i \in \{1, ..., n\}$: $(a : C, I, -) \in Trace(a : C_i, I_i, -)$ *and* $(a : C_i, I_i, -) \notin Trace(e)$ *and there is no concept* C'' *with the same properties such that* $(a : C'', I'', -) \in Trace(a : C_i, I, -)$. *If there is no such* C' *then* $C' := \top$.

Proof.

1. $C \doteq \geq mR$ can conflict only with a concept description $\leq nR$, $n < m$. Since general axioms are not allowed in our DL, the concept description $\geq mR$ has no other subsumers except $\geq kR, k < m$. Therefore $\geq nR$ is the only possible MNS for C.
2. $C \doteq \leq nR$ can be subsumed not only by $\leq kR, k > n$ but also by some concept description $D \doteq \exists R.D_1 \sqcap ... \sqcap \exists R.D_l$ such that $min_R(D) = k, k > n$. But the replacement of C with D will introduce new entailments not implied by $\mathcal{T}$. Therefore $\leq mR$ is the only possible MNS for C.
3. $C \doteq \leq nR$ not conflicting with any $\geq mR$ is involved in a clash only if it causes an impossible merging of R-successors of a. It means that the relational restrictions defined on a imply a number restriction $\geq kR$ which conflicts with $\leq nR$ because of $n < k$. Thus, $\leq kR$ is the MNS of C.
4. C' is conjunction of such C_i for which conditions (1)–(3) of Definition 2 hold. Let us consider every C_i. The fact that C occurs in the trace of C_i and an individual a belongs to both of these concepts guarantees that C is a subconcept of C_i. Since C_i does not occur in the trace of the clash element e, C_i does not reconstruct the same clash. C_i does not introduce new entailments or clashes not implied by $\mathcal{T}$, because C_i already occurs in $\mathcal{T}$ as an explicit subsumer of C. Finally, there is no concept C'' with the same properties, that is a subconcept of C_i. Therefore C' fulfills the conditions (1)–(3) of Definition 2 and is minimal towards subsumption. q.e.d.

Let us return to Example 1. The model tree for the concept *Penguin* in this example can be specified as follows:

$$\begin{aligned}\mathcal{L}(x) = \{&(a : Penguin, \varnothing, nil), (a : Bird \sqcap \neg CanFly, \{4\}, a : Penguin),\\ &(a : Bird, \{4\}, a : Bird \sqcap \neg CanFly), (a : \neg CanFly, \{4\}, a : Bird \sqcap \neg CanFly),\\ &(a : CanFly, \{4, 1\}, a : Bird), (a : CanMove, \{4, 1, 2\}, a : CanFly)\}\\ Clashes_{\mathcal{L}(x)}(Penguin) = \{&(a : \neg CanFly, \{4\}, a : Bird \sqcap \neg CanFly),\\ &(a : CanFly, \{4, 1\}, a : Bird)\}\end{aligned}$$

Thus, the set of problematic axioms is $\{1, 4\}$. Suppose the concept $CanFly$ is chosen to be removed from axiom 1. According to Lemma 2, $CanMove$ is MNS of $CanFly$. If $CanFly$ will be replaced with $CanMove$ in axiom 1, then the entailments of the form $X \sqsubseteq CanFly$, where X is a subconcept $Bird$ (for example, $Canary$) will be lost. Such situations are undesirable, because the clash

$CanFly, \neg CanFly$ concerns only the conflict between the overgeneralized concept $Bird$ and the exception $Penguin$. In order to keep the entailments we suggest to introduce to the terminology a new concept $FlyingBird$ which will capture the original meaning of $Bird$. The changed terminology looks as follows:

1. $Bird \sqsubseteq CanMove$
2. $CanFly \sqsubseteq CanMove$
3. $Canary \sqsubseteq FlyingBird$
4. $Penguin \sqsubseteq Bird \sqcap \neg CanFly$
5. $FlyingBird \sqsubseteq Bird \sqcap CanFly$

The idea of introducing a new concept A^{new} that will capture the original semantics of a rewritten overgeneralized concept A and replacing A with A^{new} in the definitions of all concepts except of the problematic one (concept $Penguin$ in Example 1) was first expressed in [OK1] for $\mathcal{ALE}$-DL. We extend this idea to $\mathcal{ALCN}$-DL.

Suppose an axiom $A \sqsubseteq D$ (or $A \equiv D$) was rewritten as $A \sqsubseteq D^*$ (or $A \equiv D^*$ respectively). We want to add to the TBox a new axiom $A^{new} \sqsubseteq D'$ (or $A^{new} \equiv D'$ respectively) such that $D' \doteq A \sqcap X$ and $X \sqcap D^* \equiv D$. In order to find X we adapt the notion of *difference of concepts* introduced in [La1].

Definition 3 *(Difference of Concepts)*
Given a TBox $\mathcal{T}$, let C and C' be concept descriptions such that $\mathcal{T} \models C \sqsubseteq C'$. The difference of C and C' is defined as different$(C, C') = \neg C' \sqcup C$.

5.3 Debugging Algorithm

Algorithm 1 constructs model trees for every atomic concept A in a terminology $\mathcal{T}$. Model trees that contain clashes are used for ontology repairing. Trees that are constructed without the merging of relational restrictions are handled first. In every equivalence class $EquivTrees$ of model trees that are constructed due to some at-most number restriction, a model tree to be repaired is chosen according to some ranking (see subsection 5.2). The trees that do not contain a clash where the given number restriction is involved have priority.

The **repair** procedure repairs the terminology using the information obtained from a model tree. For every clash (x, E), a concept description C from the trace of some element e in E is chosen to be rewritten in an axiom α_i according to a ranking. The minimal nonconflicting subsumer C' is constructed for C. C is replaced with C' in α_i. If $A_i \neq A$ then we face an overgeneralized concept. In this case, a new concept A^{new} that is supposed to capture the original semantics of A_i is added to the terminology. The name A^{new} is constructed automatically from the original name A_i and the problematic concept C. A_i is replaced with A^{new} in all axioms of $\mathcal{T}$ except α_i and the axiom defining the immediate subconcept of A_i involved in the clash e.

The **split&replace** procedure splits atomic subconcepts of A_i that are involved in the clash e except for A. Concepts appearing in the trace of e earlier are splitted first. If a definition of B_1 was rewritten and B_1 was splitted into B_1^{new} and B_1 and concept B_2 is going to be splitted immediately after B_1, then B_1 in the definition of a new concept name B_2^{new} is replaced with B_1^{new}.

Algorithm 1 Debug overgeneralized concepts in a terminology $\mathcal{T}$

$FixedClashes := \varnothing$
for all atomic concepts A such that $A \sqsubseteq D \in \mathcal{T}$ or $A \equiv D \in \mathcal{T}$
  construct model trees for A according to the expansion rules in Table 1
  for all $(x, nil) \in EquivTrees$ **repair**$(\mathcal{L}(x), A)$ **end for**
  for all $(a :\leq nR, I, a' : C') \in EquivTrees$ such that $(-, (a :\leq nR, I, a' : C')) \in EquivTrees$
    $S := \{x \mid \exists (x, (a :\leq nR, I, a' : C')) \in EquivTrees\}$
    $S_0 := \{x \in S \mid \neg\exists (x, \{(a :\leq nR, I, a' : C'), -\}) \in Clashes\}$
    if $S_0 = \varnothing$ **then**
      choose one $x \in S$ according to *ranking* and **repair**$(\mathcal{L}(x), A)$
    end if
    else choose one $x \in S_0$ according to *ranking* and **repair**$(\mathcal{L}(x), A)$
  end for
end for

Procedure **repair**$(\mathcal{L}(x), A)$
for all $(x, E) \in Clashes$ such that $E \notin FixedClashes$
  add E to $FixedClashes$
  choose an $\alpha_i \in \mathcal{T}$ ($\alpha_i = A_i \sqsubseteq C_i$ or $\alpha_i = A_i \equiv C_i$) and
    a concept C occurring in C_i such that
    $\exists e \in E :\ (a : C, \{i, -\}, -) \in Trace(e)$
    to be rewritten according to *ranking*
  $C' :=$MNS(C), C^{new} is obtained by replacement of C with C' in C_i
  replace C_i with C^{new} in α_i
  if $A_i \neq A$ **then**
    if $\alpha_i = A_i \sqsubseteq C^{new}$ **then** add $A^{new} \sqsubseteq A_i \sqcap$ *different*(C_i, C^{new}) to $\mathcal{T}$
    else add $A^{new} \equiv A_i \sqcap$ *different*(C_i, C^{new}) to $\mathcal{T}$,
      where A^{new} is a new concept name
    $T :=< e_1, ..., e_n) >$ such that
      T is a subsequence of $Trace(e)$ and
      $(a : A_i, -, -), (a : A, -, -) \notin T$ and
      $\forall k \in \{1, ..., n\} : \exists j \in \mathbb{N} :$ ($e_k = (a : A_j, I_k, -)$ and
        $A_j \sqsubseteq C_j \in \mathcal{T}$ or $A_j \equiv C_j \in \mathcal{T}$) or
        ($e_k = (a : \neg A_j, I_k, -)$ and $A_j \equiv C_j \in \mathcal{T}$) and
        $(a : A, I, -) \in E \rightarrow I \subset I_k$
    for all $\alpha \in \mathcal{T}$ such that $\alpha \neq \alpha_i$ and $e_1 = (a : A_j, -, -) \rightarrow \alpha \neq \alpha_j$
      replace A_i with A^{new} in α
    end for
    split&replace(A_i, A^{new}, T)
  end if
end for

Procedure **split&replace**(A_i, A', T)
$(a : A_j, -, -)$ is the **next** element of T
if $A_j \sqsubseteq C_j \in \mathcal{T}$ **then** add $\alpha^{new} = A^{new} \sqsubseteq C_j$ to $\mathcal{T}$,
else add $\alpha^{new} = A^{new} \equiv C_j$ to $\mathcal{T}$, where A^{new} is a new concept name
replace A_i with A' in C_j in α^{new}
for all axioms $\alpha \in \mathcal{T}, \alpha \neq \alpha_i$, $\alpha \neq \alpha_j$, replace A_j with A^{new} **end for**
split&replace(A_j, A^{new}, T)

Example 4 shows the application of the algorithm to an extended penguin example. The concept $UsedForFlying$ in axiom 1 is chosen to be rewritten. It is easy to see that the proposed algorithm extends the semantics of the "splitted" concepts, whereas the semantics of other concepts remains unchanged (for an explanation see [OK2]).

Example 4
Original terminology:
1. $Bird \sqsubseteq \exists hasPart.Wing$
2. $Wing \sqsubseteq BodyPart \sqcap UsedForFlying$
3. $Canary \sqsubseteq Bird$
4. $Penguin \sqsubseteq Bird \sqcap \forall hasPart.\neg UsedForFlying$

Changed terminology:
1. $Bird \sqsubseteq \exists hasPart.Wing$
2. $Wing \sqsubseteq BodyPart$
3. $Canary \sqsubseteq HasPartUsedForFlyingWingBird$
4. $Penguin \sqsubseteq Bird \sqcap \forall hasPart.\neg UsedForFlying$
5. $UsedForFlyingWing \sqsubseteq Wing \sqcap UsedForFlying$
6. $HasPartUsedForFlyingWingBird \sqsubseteq Bird \sqcap \exists hasPart.UsedForFlyingWing$

6 Open Problems

Several pragmatic problems of concept regeneralization have been still unsolved. Probably the most important one is the choice of the concept to be rewritten. Although there exist many axiom ranking strategies that are proposed in the literature on ontology debugging (see section 5), it is too simple to model the situation where none of these rankings except of the manual one will work. Rankings based on external knowledge seem to give better results (see [Ha1]) then pure logical approaches. An interesting task would be to compare different rankings systematically in diverse contexts.

Another interesting problem is connected with the treatment of disjunction. Suppose a concept A, $A \sqsubseteq D_1 \sqcup D_2 \in \mathcal{T}$ is overgeneralized, i.e. there exists a subconcept of A that conflicts with both D_1 and D_2. Some approaches debug one of the alternatives (see for example [La1]). Other approaches suggest to rewrite all alternatives (see [Wa1]). For both of these solutions, it is possible to model a situation where they will be relevant from a pragmatic point of view. There exists also the third possibility that has been, as far as we know, not considered in the literature yet.

Example 5 *1.* $Football\text{-}player \sqsubseteq Forward \sqcup Full\text{-}back$
2. $Goalkeeper \sqsubseteq Football\text{-}player \sqcap \neg Forward \sqcap \neg Full\text{-}back$

In Example 5, the most reasonable solution is to add *Goalkeeper* as the third alternative to the definition of $Football\text{-}player$. The problem of choosing an appropriate candidate for rewriting disjunctions is still a subject of investigation.

There exists a problem of the treatment of several overgeneralized definitions of the same concept. Consider the following example.

Example 6 *1.* $Tomato \sqsubseteq \forall hasColor.Red$ *2.* $Tomato \sqsubseteq \forall hasColor.Yellow$
3. $Red \sqsubseteq Color \sqcap \neg Yellow$ *4.* $Yellow \sqsubseteq Color \sqcap \neg Red$

In Example 6, axioms 1 and 2 contain conflicting overgeneralized definitions of the same concept. Instead of rewriting one of them we would rather replace them with their least common subsumer $\forall hasColor.Color$. The resulting axiom $Tomato \sqsubseteq \forall hasColor.Color$ would claim that the color of every tomato is a subconcept of *Color* without specifying it. Cases of multiple overgeneralization are considered in [OK2]. In the future we will adapt these ideas for $\mathcal{ALCN}$-DL.

As is easy to see, the result of the application of our algorithm depends on the order of the concepts that are input into the debugger. Unsatisfiable (or overgeneralized) concepts can be divided into two classes: (1) *Root classes* – atomic classes in which a clash or contradiction found in the class definition (axioms) does not depend on a clash or contradiction of another atomic class in the ontology and (2) *Derived classes* – atomic classes in which a clash or contradiction found in a class definition either directly (via explicit assertions) or indirectly (via inferences) depends on a clash or contradiction of another atomic class (see [Ka1]). In order to debug a derived class it is enough to debug corresponding root classes. Thus, it is reasonable to debug only the root classes. The technique of distinguishing between root and derived classes was proposed in [Ka1]. In the future we plan to integrate this technique into our approach.

7 Conclusion

We presented an approach to resolving terminological unsatisfiability with the main focus on overgeneralization. This approach is an integration of the ideas proposed in [La1] and [OK1] extended for the $\mathcal{ALCN}$ description logic. Our algorithm detects problematic axioms that cause a contradiction and automatically repairs the terminology by rewriting parts of the axioms that are responsible for the contradiction and splitting a detected overgeneralized concept into a more general and a more specific ones. This approach is knowledge preserving in the sense that it keeps as much entailments implied by the original terminology as possible.

In [Ov1], a prototypical implementation of the idea of splitting overgeneralized concepts in $\mathcal{ALE}$-DL was discussed. This implementation was tested on the famous wine-ontology[8] that was automatically extended with new classes extracted from a text corpora with the help of the Text2Onto[9] tool. Several cases of overgeneralization were detected and correctly resolved[10].

[8] `http://www.w3.org/TR/owl-guide/wine.owl`

[9] `http://ontoware.org/projects/text2onto/`

[10] For example, the class *LateHarvest* originally defined to be a sweet wine was claimed to be overgeneralized after an exception *RieslingSpaetlese* which was defined to be a late harvest wine and a dry wine appeared.

This is the first step in a more thorough evaluation. We now intend to test our approach on larger sets of axioms formalized in more expressive description logics, hereby finding out about typical problems that can arise with this kind of regeneralization procedure. We finally hope that by testing on larger and more complex real-world data we get a better understanding on what types of unsatisfiability usually occur and how frequent they are. This will help to find more application-oriented solutions to the debugging problem.

References

[Ba1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D. Patel-Schneider (eds.), P.: *Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press (2003)

[BH1] Baader, F., Hollunder, B.; Embedding defaults into terminological knowledge representation formalisms. *J. Autom. Reasoning*, 14(1) (1995) 149–180

[BS1] Baader, F., Sattler, U.: An Overview of Tableau Algorithms for Description Logics. *Studia Logica* (2001) 69:5-40

[Br1] Brandt, S., Küsters, R., Turhan, A.-Y.: Approximating ALCN-Concept Descriptions. *Proc. of the 2002 International Workshop on Description Logics* (2002)

[Fa1] Fanizzi, N., Ferilli, S., Iannone, L., Palmisano, I., Semeraro, G.: Downward Refinement in the ALN Description Logic. *Proc. of the Fourth International Conference on Hybrid Intelligent Systems* (HIS'04) (2005) 68–73

[Ha1] Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., Sure, Y.: A framework for handling inconsistency in changing ontologies. *Proc.of the Fourth Internation Semantic Web Conference*, LNCS. Springer (2005)

[Ka1] Kalyanpur, A.: *Debugging and Repair of OWL Ontologies.* Ph.D. Dissertation, University of Maryland College Park (2006)

[La1] Lam, J., Pan, J., Sleeman, D, Vasconcelos, W.: A Fine-Grained Approach to Resolving Unsatisfiable Ontologies. *Proc. of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence* (2006) 428–434

[OK1] Ovchinnikova, E., Kühnberger, K.-U.: Adaptive ALE-TBox for Extending Terminological Knowledge. In A. Sattar, B. H. Kang (eds.): *Proc. of the 19th ACS Australian Joint Conference on Artificial Intelligence*, LNAI 4304, Springer (2006) 1111–1115

[OK2] Ovchinnikova, E., Kühnberger, K.-U.: Aspects of Automatic Ontology Extension: Adapting and Regeneralizing Dynamic Updates. in M. Orgun and T. Meyer (eds.): Advances in Ontologies, *Conferences in Research and Practice in Information Technology*, 72:52–60.

[Ov1] Ovchinnikova, E., Wandmacher, T., Kühnberger, K.-U.: Solving Terminological Inconsistency Problems in Ontology Design. To appear in *International Journal of Interoperability in Business Information Systems* (IBIS) (2007)

[SC1] Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. *Proc. of IJCAI'03*, Morgan Kaufmann (2003)

[Wa1] Wang, H., Horridge, M., Rector, A., Drummond, N., Seidenberg, J.: Debugging OWL-DL Ontologies: A Heuristic Approach. *Proc. of the 4th International Semantic Web Conference 2005*, LNCS 3729 (2005) 745–757