Root Justifications for Ontology Repair

Kodylan Moodley[1,2], Thomas Meyer[1,2], and Ivan José Varzinczak[1,2]

[1] CSIR Meraka Institute, Pretoria, South Africa
{kmoodley, tmeyer, ivarzinczak}@csir.co.za
[2] School of Computer Science, University of KwaZulu-Natal, South Africa

1 Introduction

An *ontology* (also referred to as a *terminology*, *knowledge base*) is an entity used to represent some domain (field of knowledge). Usually the building blocks of an ontology include categories (concepts), relations (roles) and objects (individuals).

Description Logics (DLs) are an appropriate class of knowledge representation languages to formalize and reason about ontologies [1]. The reasoning process is carried out by a chosen *DL reasoner*. We don't provide a comprehensive introduction to DLs, but point the reader to the book by Baader et al. [1].

For our purposes a DL *TBox* consists of a finite set of *axioms* specifying the terminological part of an ontology. The Tbox includes (but need not be limited to) *subsumption statements* of the form $E \sqsubseteq F$ where E and F are (possibly complex) concept descriptions, built up from basic concepts. The semantics of DLs is based on the classical model theory for first-order logic. A DL interpretation I contains a non-empty *domain* Δ^I of elements and a mapping which interprets a basic concept A as a subset A^I of Δ^I. For purposes of illustration we shall assume that complex concepts can be constructed using negation ($\neg E$) and conjunction ($E \sqcap F$), where $\neg E$ is interpreted as $\Delta^I \setminus E^I$, and $E \sqcap F$ is interpreted as $E^I \cap F^I$. However, the inclusion of negation and conjunction is not a requirement. There may be other ways of constructing complex concepts.

An interpretation I is a *model* of a Tbox axiom $E \sqsubseteq F$ if and only if $E^I \subseteq F^I$. Given a Tbox Γ, a subsumption statement $E \sqsubseteq F$, and a basic concept A, *(i)* Γ is A-*unsatisfiable* if and only if for all models I of Γ, $A^I = \emptyset$, and *(ii)* $E \sqsubseteq F$ is a *consequence* of Γ if and only if every model of all axioms in Γ is also a model of $E \sqsubseteq F$.

Quite often during the development of ontologies, ontology developers make modelling errors. These errors can introduce *unwanted consequences* in the ontology. The process of identifying, explaining and eliminating these unwanted consequences is known as *ontology debugging and repair* or simply *ontology repair*.

In this paper, we focus on the *Black-box* approach to ontology repair which treats the reasoner as a "black-box" which basically answers 'yes' or 'no' to the question: does consequence α follow from the ontology? We extend the principles introduced in a previous work [6]. We present an improved implementation of the repair method discussed there and we also provide some experimental results comparing the performance of our approach to that of the standard *Black-box* approach to ontology repair. The standard approach has been applied mostly to the specific ontology errors called *unsatisfiable concepts* but have not been extended for other errors.

2 Ontology Debugging and Repair

In this section we discuss the standard Black-box technique for eliminating errors in the ontology based on existing work.

2.1 Debugging

Explanation is a service which focuses on explaining *why* selected consequences follow from an ontology. Given an ontology $\mathcal{O}$ with some axiom α such that $\mathcal{O} \models \alpha$, there exists an explanation which indicates *why* this is the case. The most widely used explanation is a set of *justifications* (also known as *minAs* [2] and *MUPSes* [10]) for the consequence. A justification for $\mathcal{O} \models \alpha$ is a minimal subset of $\mathcal{O}$ from which α logically follows [10].

Example 1. Consider the following ontology:

$$\mathcal{O} = \big\{ 1.\ \mathsf{C} \sqsubseteq \mathsf{A}\ 2.\ \mathsf{C} \sqsubseteq \neg\mathsf{A}\ 3.\ \mathsf{F} \sqsubseteq \mathsf{C} \sqcap \neg\mathsf{A}\ 4.\ \mathsf{F} \sqsubseteq \mathsf{C} \big\}$$

We represent $\mathcal{O}$ as the set $\{1, 2, 3, 4\}$ with the understanding that each number represents an axiom in $\mathcal{O}$. One can see that $\mathcal{O}$ is F-unsatisfiable: $\mathcal{O} \models F \sqsubseteq \bot$. Two justifications for this entailment are $\{1, 3\}$ and $\{1, 2, 4\}$ □

Justifications are useful because they allow for the pinpointing of the *causes* of modelling errors. In Example 1 they show that Axioms 1 and 3 may not both occur in $\mathcal{O}$ without $\mathcal{O}$ being F-unsatisfiable. Similarly, for Axioms 1, 2, and 4.

An example of a basic Black-box algorithm for computing a justification for an ontology consequence is the *naïve pruning algorithm* [11]. It works by removing axioms from the ontology, one at a time, and monitoring how this affects the entailment under consideration. For details of this algorithm the reader should consult the provided reference.

In order to obtain a *complete* explanation for the unsatisfiability of a concept one has to determine *all* its justifications. The most well known method to do this is a variant of *Reiter's [9] hitting set algorithm.* This variant algorithm [7] generates a *justification tree* for the unsatisfiability of the concept w.r.t. $\mathcal{O}$. For details of this method the reader should consult the reference provided.

2.2 Repair

Recall Example 1. One can consider justifications $\{1, 3\}$ and $\{1, 2, 4\}$ as reasons for the F-unsatisfiability of $\mathcal{O}$. To eliminate the unsatisfiability, one has to nullify *all* its reasons. A common strategy to do this is to remove a single axiom from each justification for the unsatisfiability [4].

Therefore, if we remove the set $\{2, 3\}$ from $\mathcal{O}$ then the unsatisfiability of F is eliminated. The resulting ontology $\mathcal{O}\backslash\{2, 3\}$ is a *repair* [6] for the F-unsatisfiability of $\mathcal{O}$ and the set $\{2, 3\}$ is a *diagnosis* [9] for the F-unsatisfiability of $\mathcal{O}$. A key requirement of computing repairs (diagnoses) is to find *maximal* repairs (*minimal* diagnoses).

In the case of a set of unsatisfiable concepts, $\mathcal{C}$, in some ontology $\mathcal{O}$, the goal is to repair $\mathcal{O}$ by replacing it with an ontology $\mathcal{O}'$ which is C*-satisfiable* for every $C \in \mathcal{C}$.

Kalyanpur et al. [5] discuss an approach for eliminating this set of unsatisfiable concepts in the ontology. This approach separates the unsatisfiable concepts into *root* unsatisfiable concepts and *derived* unsatisfiable concepts. Intuitively, a root unsatisfiable concept is a concept whose unsatisfiability is not caused by that of another concept in the ontology. A derived unsatisfiable concept is one which is *not* root unsatisfiable.

A useful property of a root unsatisfiable concept, C, is that if one repairs the unsatisfiability of C then all other concepts in the ontology whose unsatisfiability is caused by that of C are automatically repaired [5]. Using this property, one can resolve a set of unsatisfiable concepts in some ontology as follows. Initially one identifies all the root unsatisfiable concepts in the set. The unsatisfiabilities of these concepts are then eliminated by removing an axiom from each of their justifications from the ontology. After this, one has to re-compute/re-identify which of the *remaining* concepts in the set are root unsatisfiable (some concepts which were derived unsatisfiable may become root unsatisfiable after the initial repair).

This process has to be repeated until there are no more unsatisfiable concepts in the set. The drawbacks to this repair strategy are that (*i*) it is only applicable to one type of ontology error (unsatisfiable concepts) and (*ii*) it does not eliminate the entire set of unsatisfiable concepts simultaneously. Rather, it uses an iterative approach as described above.

3 Repair using Root Justifications

In this section, we discuss our approach to ontology repair. This approach can be applied to a set of *unwanted axioms*. We start by observing that the most prominent types of ontology error (unsatisfiable concepts and ontology inconsistency) can be generalized to some unwanted axiom [7]. Thus, the key differences from the method discussed in the previous section are (*i*) We are dealing with *other* types of errors as well, not just unsatisfiable concepts and (*ii*) We are eliminating a *set* of such errors *simultaneously*.

We first define some terminology that will be used in the remainder of this section. Given an ontology $\mathcal{O}$ and and some unwanted axiom α_U, a justification J for $\mathcal{O} \models \alpha_U$ is a α_U*-justification* for $\mathcal{O}$. The set of all α_U-justifications for $\mathcal{O}$ is denoted by $\mathcal{J}_{\mathcal{O}}(\alpha_U)$. Given an ontology $\mathcal{O}$ and a *set* of unwanted axioms $\mathcal{U}$, the set $\mathcal{J}_{\mathcal{O}}(\mathcal{U}) = \bigcup_{\alpha_U \in \mathcal{U}} \mathcal{J}_{\mathcal{O}}(\alpha_U)$. Using this terminology we characterize a root justification as follows.

Definition 1 ($\mathcal{U}$-root justification). *Given an ontology $\mathcal{O}$ and a* set *of unwanted axioms $\mathcal{U}$, a set RJ is a $\mathcal{U}$-*root justification *for $\mathcal{O}$ if and only if it is a α_U-justification for $\mathcal{O}$ for some $\alpha_U \in \mathcal{U}$ (i.e. $RJ \in \mathcal{J}_{\mathcal{O}}(\mathcal{U})$), and there is no $J \in \mathcal{J}_{\mathcal{O}}(\mathcal{U})$ such that $J \subset RJ$. We denote the set of all $\mathcal{U}$-root justifications for $\mathcal{O}$ by $\mathcal{RJ}_{\mathcal{O}}(\mathcal{U})$.*

In the work by Kalyanpur et al. [5] on root and derived unsatisfiable concepts, root justifications are used (implicitly) as a means to identify the root and derived unsatisfiable concepts. For our approach to ontology repair, the notion of a root justification is central and thus we highlight this principle here.

Example 2. For Example 1, let $\mathcal{U} = \{F \sqsubseteq \bot, C \sqsubseteq \bot\}$. We have already seen that $\mathcal{J}_{\mathcal{O}}(F \sqsubseteq \bot) = \{\{1,3\},\{1,2,4\}\}$. It is easy to see that $\mathcal{J}_{\mathcal{O}}(C \sqsubseteq \bot) = \{\{1,2\}\}$ and therefore that $\mathcal{J}_{\mathcal{O}}(\mathcal{U}) = \{\{1,3\},\{1,2,4\},\{1,2\}\}$. Therefore, according to Definition 1, the set of $\mathcal{U}$-root justifications for $\mathcal{O}$ is $\mathcal{RJ}_{\mathcal{O}}(\mathcal{U}) = \{\{1,2\},\{1,3\}\}$. □

The following algorithm demonstrates the computation of a *single* root justification $RJ \in \mathcal{RJ}_{\mathcal{O}}(\mathcal{U})$:

Algorithm 1: (Single root justification)

```
  Input: Ontology $\mathcal{O}$, unwanted axiom set $\mathcal{U}$ ($|\mathcal{U}| \geq 1$)
  Output: $\mathcal{U}$-root justification, $RJ$, for $\mathcal{O}$
  Uses: entailedAxioms($\mathcal{O}, \mathcal{U}$), which returns $\{\alpha \in \mathcal{U} \mid \mathcal{O} \models \alpha\}$
1 $RJ := \mathcal{O}$;
2 foreach $\alpha \in RJ$ do
3     if $|entailedAxioms(RJ \backslash \{\alpha\}, \mathcal{U})| \geq 1$ then
4         $RJ := RJ \backslash \{\alpha\}$;
5     end
6 end
7 return $RJ$;
```

The key difference between Algorithm 1 and the naïve pruning algorithm is that we are now considering the entailment of *all* unwanted axioms in a set (in procedure *entailedAxioms*(.)), rather than just a single axiom. It is clear that Algorithm 1 terminates for all finite inputs of $\mathcal{O}$ and $\mathcal{U}$. This follows from Line 2 of the algorithm which shows that the loop only considers the axioms in the finite set RJ.

Algorithm 1 is computationally intensive because we only consider a single axiom at a time in the ontology and for each consideration we require between 1 and $|\mathcal{U}|$ entailment tests. We use a more optimized version of this algorithm in practice (based on a sliding window technique [4]). Details of this algorithm [7] can be found in the reference provided.

In order to ensure that we are able to generate *all* the repairs for an unwanted axiom set in an ontology, it is necessary to know *all* the root justifications for the unwanted axiom set. We have given a brief description of a variant of Reiter's Hitting Set Algorithm which computes *all* the "regular" justifications for a single entailment [4]. This variant algorithm can also be used (with some slight modifications) to compute all *root* justifications for a *set* of unwanted axioms. The algorithm [7] can be found in the reference provided. The significance of root justifications is that they can be used to generate precisely the $\mathcal{U}$-repairs for $\mathcal{O}$.

Definition 2 ($\mathcal{U}$-repair). *A subset R of $\mathcal{O}$ is a $\mathcal{U}$-*repair *for $\mathcal{O}$ if and only if $R \not\models \alpha_U$ for every $\alpha_U \in \mathcal{U}$, and for every R' for which $R \subset R' \subseteq \mathcal{O}$, $R' \models \alpha_U$ for some $\alpha_U \in \mathcal{U}$.*

We denote the set of $\mathcal{U}$-repairs for $\mathcal{O}$ by $\mathcal{R}_{\mathcal{O}}(\mathcal{U})$. For Example 1 it can be verified that $\mathcal{RJ}_{\mathcal{O}}(\{\mathsf{C} \sqsubseteq \bot, \mathsf{F} \sqsubseteq \bot\}) = \{\{1,3\},\{1,2\}\}$ and thus $\mathcal{R}_{\mathcal{O}}(\{\mathsf{C} \sqsubseteq \bot, \mathsf{F} \sqsubseteq \bot\}) = \{\{2,3,4\},\{1,4\}\}$. The above $\mathcal{U}$-repairs can be generated from *$\mathcal{U}$-diagnoses.*

Definition 3 ($\mathcal{U}$-diagnosis). *A subset D of $\mathcal{O}$ is a $\mathcal{U}$-*diagnosis *for $\mathcal{O}$ if and only if $D \cap RJ \neq \emptyset$ for every $RJ \in \mathcal{RJ}_{\mathcal{O}}(\mathcal{U})$. D is a* minimal *$\mathcal{U}$-diagnosis for $\mathcal{O}$ if and only if there is no $\mathcal{U}$-diagnosis D' (for $\mathcal{O}$) such that $D' \subset D$.*

4 Implementation and Evaluation

We have implemented a Protégé 4 plugin [3] for computing root justifications for sets of unwanted axioms. We have extended it to also compute the $\mathcal{U}$-repairs. We have performed some preliminary experiments to compare this approach for ontology repair with the naïve sequential approach described in Sect. 2.2.

We use three sample ontologies (The Tambis [3], Travel [4] and Pizza [8] ontologies) of varying size, structure, and application. These ontologies are expressed in the DLs $\mathcal{SHIN}$, $\mathcal{SOIN}(\mathcal{D})$ and $\mathcal{SHOIN}$ respectively. There are three test cases, one for each ontology and in each case we select four different unwanted axiom sets from the ontology. Each test case has four experiments (one for each unwanted axiom set). In each experiment, we perform a *control* computation. This control computation uses the *naïve approach* to identify all *regular* justifications for each axiom in the unwanted axiom list. We then use our approach to identify all *root* justifications for the unwanted axiom set. The performance of these two approaches is then compared. For details about the results of these experiments [7] the reader should consult the given reference.

To summarize these results, the performance of *OntoRepair* depends on the kinds of axioms contained in the unwanted axiom set. This is because the specific axioms in the set influence the *justifications* which are computed and thus also the number of root justifications for the set versus the number of regular justifications for each unwanted axiom in the set. Nine out of twelve of our experiments

[3] `http://krr.meraka.org.za/software/ontorepair`
[4] `http://protege.cim3.net/file/pub/ontologies/travel/travel.owl`

show instances where the number of root justifications is less than the number of regular justifications. This frequency justifies the use of *OntoRepair* which provides improved performance (over the naïve approach) in all these instances. However, there is scope for optimizing the computation of root justifications in *OntoRepair* to be comparable to the performance of the naïve approach, for those cases in which the number of root and regular justifications are equal. Finally, it is important to mention that since the empirical analysis was conducted on the basis of just three examples, the results are unlikely to be statistically significant and conclusive.

5 Conclusion

We have presented an alternative approach for ontology repair using the notion of root justifications. We have implemented a Protégé 4 plugin, *OntoRepair*, to demonstrate this repair strategy. Some preliminary experiments show overall better performance than a naïve approach to ontology repair. However, there are some special cases in which the naïve approach performs far better than our approach and vice versa. More experiments will be performed to characterize the circumstances in which our approach gains a clear advantage. Various performance optimizations for the *OntoRepair* tool are also in the pipeline.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge, 2 edition, 2007.

[2] F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the Description Logic $\mathcal{EL}$. In *Proc. KI*. Springer, 2007.

[3] P. G. Baker, C. A. Goble, S. Bechhofer, N. W. Paton, R. Stevens, and A. Brass. An ontology for bioinformatics applications. *Bioinformatics*, 15(6):510–520, 1999.

[4] A. Kalyanpur. *Debugging and repair of OWL ontologies*. PhD thesis, University of Maryland, 2006.

[5] A. Kalyanpur, B. Parsia, E. Sirin, and J. Hendler. Debugging unsatisfiable classes in OWL ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(4):268–293, 2005.

[6] T. Meyer, K. Moodley, and I. Varzinczak. First steps in the computation of root justifications. In *Proc. ARCOE*, 2010.

[7] K. Moodley. Debugging and repair of Description Logic ontologies. Master's thesis, University of KwaZulu-Natal, 2011.

[8] A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In *Engineering Knowledge in the Age of the SemanticWeb*, volume 3257. 2004.

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

[10] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. IJCAI*, 2003.

[11] B. Suntisrivaraporn, G. Qi, Q. Ji, and P. Haase. A modularization-based approach to finding all justifications for OWL DL entailments. In *Proc. ASWC*, 2008.