Universität Koblenz-Landau

Proposal for a master's thesis

How I forgot my OWL
and where I found it again

Holger Johannes Heinz
(MatNr. 209210903, hheinz@uni-koblenz.de)

supervised by
Professor Dr. Steffen Staab
Dr. Claudia Schon

Abstract

Ontologies are an important part of the semantic web and essential tools for knowledge representation. They can be used to describe different concepts and their relations through well-defined building blocks. This allows machines to process the represented knowledge in order to answer queries and derive new statements. Since ontologies undergo continuous change it is important to have tools at hand that support this process. Changes in the domain knowledge or modeling errors introduced by human editors require a modification of the axioms that describe a concept. Furthermore, statements that were once removed could eventually find their way back into the ontology. As Description Logics provide the formal ground for reasoning in ontologies such as OWL DL, there is a need for well-defined formal methods that can be used to express these changes.
In the following, I therefore propose a topic for my master's thesis: The development of formal methods that support a fine-grained and reversible removal of conjuncts from $\mathcal{EL}$ axioms, based on axiom labeling and on the work of Suchanek et al. [1].

1 Context and Motivation

This section gives an overview about the context of the proposed research topic and about the problems that are addressed by it.

1.1 Context

Since Tim Berners-Lee formulated the idea of a semantic web in 2001 [2], there has been much growth in this area of research. Different projects such as DBPedia[1] aim at giving formerly unstructured knowledge a well-defined, ontology based structure which enables agents to process this data. An ontology contains a collection of definitions of concepts and makes it possible to establish a common terminology between these agents. By this, they can agree on the same semantic notion of a specific term.[3]
Today, ontologies are used for knowledge representation in many different areas such as medicine[2] and biology[3]. OWL[4] and its successor OWL 2[5] are a family of description languages for ontologies that are recommended by the W3C. Reasoning is an important step in the development of ontologies. It can be used to test if an ontology is free of contradictions and to support the integration of multiple ontologies in the context of interoperability.[3]
Description Logics can provide the foundation for this reasoning. An example for this are the OWL sub-languages OWL DL and OWL Lite. Their semantics correspond to the Description Logics $\mathcal{SHOIN}(D)$ in the case of OWL DL and $\mathcal{SHIF}(D)$ in the case of OWL Lite, as shown by Horrocks and Patel-Schneider [6].

[1] http://www.dbpedia.org
[2] http://www.snomed.org
[3] http://www.geneontology.org

1.2 Motivation

Ontologies evolve over time when new knowledge is added or existing knowledge is revised. During this process it can be necessary to remove some statements from the ontology, even if they do not cause inconsistencies on a logical level. To give an example, scientific research could alter our understanding of the human body and result in new medical knowledge, which ultimately would make its way into medical ontologies such as SNOMED CT. When this change is performed by human ontology editors, modelling errors can be introduced into the ontology. Cornet et al. [7] studies certain cases in which modifications of SNOMED CT lead to these problems, such as the introduction of nonsensical clinical concepts or the existance of seemingly equal concepts, which yet differ in their underlying, logical representation. If this occurs, it could be reasonable to remove certain new concepts again to recover from these errors.
Moreover, there are cases where it can be worthwhile to keep some kind of record of the statements that were removed. Think of an ontology that is used to describe some industrial good and is used in its production. Imagine then that some day in the future a part of this product must be removed because it has become technical obsolete. Consider for example the following $\mathcal{EL}$ axiom that describes a car as a vehicle with that has an ignition lock:

$$Car \equiv Vehicle \sqcap \exists hasPart.IgnitionLock$$

In the future, cars may not need an ignition lock anymore because they will be controlled by an electronic lock. If the above axiom was part of an ontology one would need to update the ontology to cope with this change. A first idea to do so would be to remove the whole concept *Car* from the ontology and to add a new formalization of a car. However, this would also remove the knowledge that a car is a vehicle. A possible better solution could be to only remove the part from the axiom that refers to the ignition lock.
Some time later one could want to undo the changes that were performed earlier. This could especially happen if modeling errors were accidentally introduced into the ontology but also if a trend emerges in product design that favors things that have previously been written-off and were already removed from the ontology, such as analog clockworks for watches, which are still in production today. If some record of the earlier change was kept, then it could be less cost-intensive to transform the concepts back into their previous state. A straight-forward solution for this would be to create a backup of the whole ontology when a change is made and to roll it back the whole ontology every time it is necessary. However, this is not efficient in terms of storage usage and would undo desired changes that were performed in the meantime.

2 Related Work

Ontology change is an broad field of research. It incorporates different areas such as ontology evolution, versioning, debugging, integration and mapping which deal with the change process from different points of view, such as detecting inconsistencies when the ontology is updated or propagating changes. With [8] an extensive survey is available.
Several approaches to keep track of ontology changes and to handle ontology

evolution exist. These methods often do not focus on reversibility of a single operation but deal with the change process in a more coarse-grained way like Klein and Noy [9], Stojanovic et al. [10]. In addition, the proposed methods for preserving information about changes sometimes compute the changes in retrospection on a structural level [11] or let the user decide in which possible interpretation of the change he is interested in [12, 13]. Furthermore, the proposed techniques that keep track of ontology changes are often part of a larger framework [12, 13, 14]. An example for such a framework is Khattak et al. [14] which contains a so called Change History Log that is used to record changes. The log is stored in a Jena driven triple store within the framework.
Furthermore, axiom labeling can be used to attach additional meta-information about changes to axioms, such as trust levels or modification times. An example for this is given by Schenk et al. [15] where axiom labels are used to provide provenance information about inferred knowledge.

3 Approach

The examples in the previous section show that it could be desirable to have a method at hand that supports a fine-grained removal of axiom conjuncts. Furthermore, this method should also support to undo previous changes.
Some operations that can perform changes at sub-axiom level are given by Suchanek et al. [1], on which I propose to base my work. This includes an operator for subtraction that can be used to remove single conjuncts from an axiom. Consider for example the concept of a car, defined as a vehicle that has a door and a window:

$$Car \equiv Vehicle \sqcap \exists hasPart.Door \sqcap \exists hasPart.Window$$

If one wants to get rid of the door, it is possible to use the subtraction operator provided by Suchanek et al. [1] to remove the conjunct $\exists hasPart.Door$ from the above concept of a car. This results in the desired concept

$$Car \equiv Vehicle \sqcap \exists hasPart.Window.$$

However, there are situations in which this operator does not yield the desired result. These are shown in the following:

Greedy removal of elements Consider again the concept of a car, but this time in a more complex form:

$$IgnitionSystem \equiv \exists hasPart.IgnitionLock \sqcap \exists hasPart.Starter$$

$$Car \equiv Vehicle \sqcap \exists hasPart.IgnitionSystem$$

Here a car is described as a vehicle that has an ignition system, which is made up of an ignition lock and an electric starter. Now assume one wants to get rid of the fact that a car has an ignition lock. One could try to subtract the statement $\exists hasPart.(\exists hasPart.IgnitionLock)$ from Car. In this case one could expect to end up with a new concept that represents the same car as above, with the only change that the car now lacks the ignition lock:

$$Car \equiv Vehicle \sqcap \exists hasPart.(\exists hasPart.Starter)$$

However, the subtraction operation returns a car that also lacks the starter for its engine, resulting in $Car \equiv Vehicle$.
One could argue that it was wrong to alter *Car* in the first place and that the subtraction should have been applied to *IgnitionSystem* instead. This would have yielded the intended result

$$IgnitionSystem \equiv \exists hasPart.Starter$$

and the ignition lock would have been removed without the immediate loss of other knowledge. However, this could have unwanted side effects. It is easy to think of a case in which the concept *IgnitionSystem* could have also been used in the definition of other vehicles. The subtraction operation would then strip the ignition lock from all these other concepts, too. This could particularly happen in a larger ontology where concepts are reused multiple times. Therefore, it would be desirable to have an operator at hand that supports the removal of knowledge in a less extensive way.

Scope-independent removal of conjuncts In addition, it could be beneficial if such an operator supports the removal of all occurrences of a certain concept from an axiom, independent of the scope of role-restrictions at which the concept is found. Consider for example the concept of a higher class car that gets an expensive pearlescent finish. The car's door as well as the moldings that are part of the door must be painted in the same way:

$$\begin{aligned} HigherClassCar \equiv\ & Vehicle \sqcap \exists hasPart.(Door \sqcap \exists hasFinish.Pearlescent \\ & \sqcap \exists hasPart.(Molding \sqcap \exists hasFinish.Pearlescent)) \end{aligned}$$

If one wants to get rid of this finish, the *Pearlescent* concept must be removed from all *hasPart*-roles, independent of the nesting depth of role restrictions it occurs under. Unfortunately, the operator proposed by Suchanek et al. [1] does not directly support this.

Inadequate reversibility It is necessary to overcome another problem in order to use the operator for the examples that are described in section 1.2. The problem is caused by the fact that the subtraction operator is reversible only in certain cases.

$$(A - B) + B = A$$

only holds if and only if every conjunct of the subtrahend B appears in the minuend A. This property makes it non-trivial to undo a subtraction operation in cases where the above constraint does not hold, even if a log of all applied operations was kept.

When considering the previous examples one can see that it is desirable to have tools at hand that allow the fine-grained removal of statements at sub-axiom level. To support reversibility in every case, it would be helpful if an ontology existed that keeps a log of all applications of this operator in such a way that

the previous stated conditions do not affect reversibility anymore. Using this ontology-based mechanism, one could also store additional meta-information about a change, such as:

- The reason why an axiom was changed,
- the logical operation that was used to change the axiom,
- the timestamp of the change,
- the name of the employee or user who initiated the change,
- and the overall order in which the changes took place.

This entries could then be used to reconstruct the nature of a change or to provide additional provenance information about an entailed consequence. Even if some of this information would be stored already in some kind of formal or non-formal documentation, this method has the additional benefit that the information is directly tied to the ontology which it references to. If the ontology is shared or moved this documentation is then passed along with it.

4 Aim & Methodology

The proposed aim of my master's thesis is the development of a method that allows a fine-grained and reversible removal of conjuncts from $\mathcal{EL}$ axioms in such a way that the problems shown in section 3 are overcome. The method shall be based on the work of Suchanek et al. [1] and on axiom labeling related to Schenk et al. [15]. This is broken-down in the following three requirements:

Requirement 1 A new operator based on the work proposed by Suchanek et al. [1] will be defined. It must have the following properties:

- The application of this operator to an axiom must remove as less conjuncts of an axiom as possible, i.e. no conjunct must be removed that does not appear in the minuend.
- The new operator must support the simultaneous removal of multiple conjuncts. It must remove conjuncts independent of the nesting depth of role restrictions under which they occur.
- The operator must be defined in terms of the description logic $\mathcal{EL}$.

Requirement 2 An integrated version-log for ontologies based on axiom labeling will be developed. The system must keep track of axiom changes and must support storing meta-information about every change, such as the time of change. The thesis must answer the question how a unlabeled ontology can be combined with a version-log, how axioms in this new ontology can be changed and how the latest version of the axioms can be obtained. Considering efficiency, the version-log must work in such a way that when an axiom is changed no duplicate of the ontology is created.

Requirement 3 A prototype implementation of this new operator shall be programmed, based on the implementation hopefully provided by the authors of [1]. The implementation shall support removing and restoring conjuncts from and to $\mathcal{EL}$ axioms utilizing the methods specified in the requirements 1–2.

The implementation will be tested using different $\mathcal{EL}$ ontologies in order to judge the practicability of the proposed approach.

References

[1] F. M. Suchanek, C. Menard, M. Bienvenu, and C. Chapellier, "Can You Imagine... A Language for Combinatorial Creativity?" in *International Semantic Web Conference*. Springer, 2016, pp. 532–548.

[2] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 28–37, 2001.

[3] F. Baader, I. Horrocks, and U. Sattler, "Description Logics as Ontology Languages for the Semantic Web," in *Mechanizing Mathematical Reasoning*. Springer, 2005, pp. 228–248.

[4] M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "OWL Web Ontology Language Reference," *W3C Recommendation February*, vol. 10, 2004, accessed on 2017-03-10. [Online]. Available: https://www.w3.org/TR/owl-ref/

[5] "OWL 2 Web Ontology Language Document Overview (Second Edition)," Dec 2012, accessed on 2017-03-10. [Online]. Available: https://www.w3.org/TR/owl2-overview/

[6] I. Horrocks and P. F. Patel-Schneider, "Reducing OWL Entailment to Description Logic Satisfiability," in *International Semantic Web Conference*. Springer, 2003, pp. 17–29.

[7] R. Cornet, M. Nyström, and D. Karlsson, "User-directed Coordination in SNOMED CT," *MedInfo*, vol. 192, pp. 72–76, 2013.

[8] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, and G. Antoniou, "Ontology Change: Classification and Survey," *The Knowledge Engineering Review*, vol. 23, no. 2, pp. 117–152, 2008.

[9] M. Klein and N. F. Noy, "A Component-based Framework for Ontology Evolution," in *Workshop on Ontologies and Distributed Systems at IJCAI*, vol. 3, 2003.

[10] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic, "User-driven Ontology Evolution Management," in *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 2002, pp. 285–300.

[11] N. F. Noy, M. A. Musen *et al.*, "Promptdiff: A Fixed-point Algorithm for Comparing Ontology Versions," *AAAI/IAAI*, vol. 2002, pp. 744–750, 2002.

[12] P. Plessers and O. De Troyer, "Ontology Change Detection Using a Version Log," in *International Semantic Web Conference*. Springer, 2005, pp. 578–592.

[13] P. Plessers, O. De Troyer, and S. Casteleyn, "Understanding Ontology Evolution: A Change Detection Approach," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 1, pp. 39–49, 2007.

[14] A. M. Khattak, K. Latif, and S. Lee, "Change Management in Evolving Web Ontologies," *Knowledge-Based Systems*, vol. 37, pp. 1–18, 2013.

[15] S. Schenk, R. Dividino, and S. Staab, "Using Provenance to Debug Changing Ontologies," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 3, pp. 284–298, 2011.