

# Towards Real Time Ontology Editing

**Christian Halaschek-Wiener**

University of Maryland, College Park, MD, USA  
Department of Computer Science  
halasche@cs.umd.edu

**Bijan Parsia**

The University of Manchester, UK  
School of Computer Science  
bparsia@cs.man.ac.uk

## Abstract

Recently there has been substantial development of OWL ontology engineering tools. Such tools typically come equipped with fully functional Description Logic (DL) reasoners, as both OWL-Lite and OWL-DL are formally aligned with Description Logics. One main limitation of editors is that the underlying reasoners cannot be continuously turned on while editing large, expressive ontologies, due to the complexities of DL reasoning and desired UI response times. In this paper, we present a variety of techniques, many of which leverage our recent work on incremental DL reasoning, that aid in achieving real time DL reasoning for ontology editors. We provide an overview of the techniques and an implementation in the ontology editor Swoop.

## 1 Introduction

Over the past few years there has been substantial development of ontology engineering tools (e.g., OntoTrack, Swoop, Protege, etc.) primarily fueled by the standardization of the Web Ontology Language (OWL). Such tools typically come equipped with fully functional Description Logic (DL) reasoners, as both OWL-Lite and OWL-DL are formally aligned with the Description Logics *SHIF* and *SHOIN*, respectively. This allows the tools to provide users with classified views of concept hierarchies, specific class types of individuals (i.e., realization), un-satisfiable concepts, etc.

Recently, we have seen very large and expressive OWL ontologies emerge in a variety of domains including life sciences, wine, finance, etc. Due to the complexities of DL reasoning and desired UI response times, most editors do not do continuous reasoning while one is editing. Rather most tools relying on an analogue of the edit-compile-test loop of most programming environments. If this cycle is very long (e.g., hundreds of seconds) then users will be forced to perform larger sets of edits before testing.

In this paper, we present a variety of techniques which provide an initial step towards continuous reasoning while editing ontologies expressed using the Description Logic *SHIQ*, which largely overlaps with OWL-DL. Many of the techniques presented leverage our recent work on incremental

reasoning through syntactic updates to knowledge bases expressed in *SHIQ*. Additionally, we present an implementation of the techniques in the OWL ontology editor Swoop [Kalyanpur *et al.*, 2005].

## 2 Reasoning Through Ontology Change

In order to provide reasoning through updates in an ontology editor, we adopt syntactic changes/updates of knowledge base assertions at both the terminological (TBox) and assertional (ABox) levels, which we refer to as *Syntactic Updates*. Intuitively, *Syntactic Updates* can be described as an update in which all new assertions are directly added (or removed) to the asserted (base) axioms. Formally, we describe this update as follows:

**Definition 1** (*Syntactic Updates*) *Let  $S$  be the set of assertions in an initial knowledge base  $K$ . Then under Syntactic Updates, updating  $K$  with an addition (resp. deletion)  $\alpha$ , written as  $K + \alpha$  (resp.  $K - \alpha$ ), results in an updated set of axioms  $S'$  such that  $S' = S \cup \{\alpha\}$  (resp.  $S' = S \setminus \{\alpha\}$ ).*

This type of update semantics is adopted as it clearly aligns with the task of reasoning through updates in an ontology editor.

### 2.1 Incremental Reasoning for Ontology Editors

#### Lazy Reasoning

One intuitive, yet extremely effective optimization for achieving more real time reasoning in ontology editors, is that only reasoning related to the current components of the ontology that are visible to the user need be performed immediately. For example, concept satisfiability needs only be immediately performed for the concepts currently viewable in the tool. Additionally, realization only needs to be immediately performed for individuals currently in view. All other reasoning can be delayed until after the reasoning services for these viewable portions of the ontology is performed. We refer to this as *lazy reasoning*.

#### Consistency Checking

After a KB is updated, consistency must be rechecked; if the KB becomes inconsistent after an update then everything is trivially entailed and the user interface of the editor must be updated. To address this we propose leveraging our recent work on incremental consistency checking of the

Description Logic *SHIQ* [Halaschek-Wiener *et al.*, 2006]. In [Halaschek-Wiener *et al.*, 2006], we present an approach for incrementally updating tableau completion graphs under ABox changes interpreted under syntactic updates, which provides a mechanism to incrementally determine if the updated KB is consistent. The approach has demonstrated orders of magnitude performance improvement and performs in real time for a variety of realistic ontologies. Due to space limitations, further details are omitted here; however they can be found in [Halaschek-Wiener *et al.*, 2006].

### Classification and Realization

As discussed earlier, we only need to classify and perform realization for concepts and individual which are currently viewable within the UI of the ontology editor. This allows us to drastically reduce the number of immediate consistency checks for classification (resp. realization), by avoiding checking subsumption relations for concepts (resp. instantiation of individuals) not visible in the tool.

In our recent work, we have additionally proposed a variety of caching techniques to reduce the number of consistency check required during classification and realization [Parsia *et al.*, 2006]. Under axiom additions, consistency checks for all previously known subsumption relations and individual concept instantiations can be avoided due to monotonicity of OWL-DL. Additionally, under additions, by caching of pseudo models of previously known non-subsumptions and non-instantiations, tests can be avoided if the pseudo models for the respective concepts and/or individuals have not changed. Under deletions, consistency checks for all previously known non-subsumption relations and individual concept non-instantiations can be avoided due to monotonicity of OWL-DL. Lastly, under deletions, previously known subsumption and concept instantiation test can be avoided by leveraging previous work on finding the justifications for inferences in OWL-DL using axiom tracing. In particular, if the removed assertion is not included in the justification for the non-subsumption then it can be assumed that the non-subsumption still holds. Further details are omitted here; however they can be found in [Parsia *et al.*, 2006].

### Concept Satisfiability

We first note that a similar caching technique can be used as mentioned above; namely under additions only previous concepts that were satisfiable can become unsatisfiable. All concepts that were unsatisfiable prior to the addition will remain as such. In contrast, under deletion only previously unsatisfiable concepts can become satisfiable. All other previously satisfiable must remain satisfiable.

An additional optimization for the purpose of concept satisfiability includes caching the tableau completion graphs from previous concept satisfiability tests. The general idea is to incrementally update the cached completion graphs using the techniques briefly discussed above [Halaschek-Wiener *et al.*, 2006]. We note that for extremely large TBoxes, this approach may not be appropriate due to memory overhead.

### Exploiting Ontology Modularity

Recently there has been work in finding modular subsets of DL ontologies [Grau *et al.*, 2006]. This work can be lever-

aged for reasoning in ontology editors as follows: in the case that edits are performed to a particular module, then reasoning only needs to be performed on that module. For large ontologies with many modules, this technique will greatly reduce reasoning time as only a small subset of the ontology must be processed.

## 3 Editing in Swoop

The previous techniques have been implemented in the ontology browser and editor Swoop [Kalyanpur *et al.*, 2005]. Figure 1 shows the main interface for Swoop, including an editing pane for adding information to the ontology.

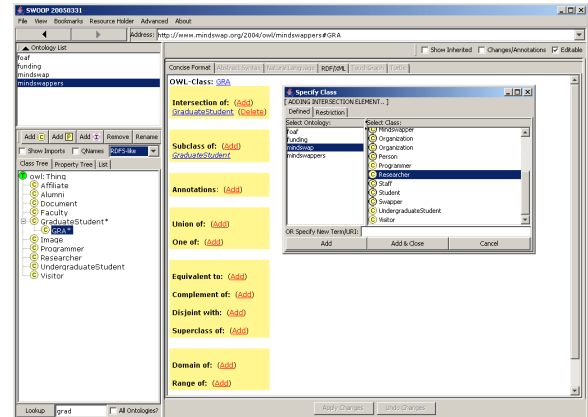


Figure 1: Editing in Swoop

## 4 Conclusion and Future Work

In this paper, we have presented a variety of techniques for the purpose of moving towards more real time reasoning in ontology editors. A variety of these approaches focus on addressing incremental reasoning through syntactic changes of ontologies. Future work includes investigating additional techniques for incremental reasoning and further investigating the performance improvements of the techniques presented in this paper.

## References

- [Grau *et al.*, 2006] Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Modularity and web ontologies. In *In Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning*, 2006.
- [Halaschek-Wiener *et al.*, 2006] Christian Halaschek-Wiener, Bijan Parsia, and Evren Sirin. Description logic reasoning with syntactic updates. In *In Proc. of the 5th Int. Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2006)*, 2006.
- [Kalyanpur *et al.*, 2005] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, and James Hendler. Swoop: A 'web' ontology editing browser. In *Journal on Web Semantics 4(2)*, 2005.
- [Parsia *et al.*, 2006] Bijan Parsia, Christian Halaschek-Wiener, and Evren Sirin. Towards incremental reasoning through updates in owl-dl. Reasoning on the Web Workshop - WWW2006, 2006.

## Demonstration Overview

A live software demonstration will be presented. In the demonstration a version of the OWL ontology editor Swoop<sup>1</sup> will be shown that includes an implementation of the techniques discussed in this paper. The main interface of Swoop is shown below in Figure 2.

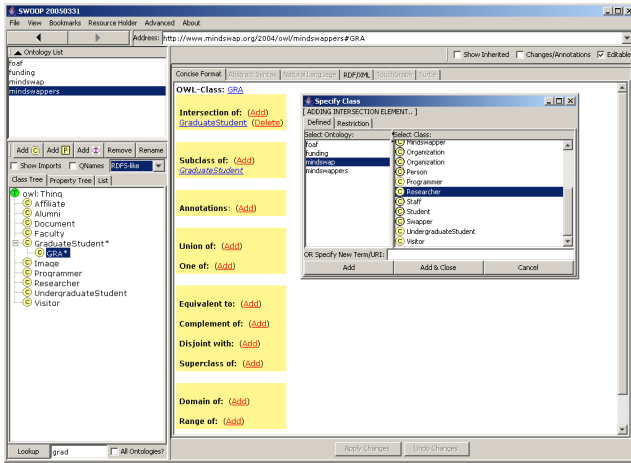


Figure 2: Swoop main user interface.

In the demonstration a variety of OWL ontologies, including a financial<sup>2</sup> ontology, will be loaded into Swoop. The Pellet<sup>3</sup> DL reasoner, which has the previously mentioned incremental reasoning optimizations built in, will be continuously turned on within Swoop. Various additions and deletions of both ABox and TBox assertions will be performed on the ontologies loaded into Swoop. In the demonstration we will show the effectiveness of the techniques discussed in this paper. In particular we will demonstrate the utility of modularity for reasoning over only a subset of the ontology. We will show the use of our incremental consistency checking algorithm by adding (resp. removing) assertions which make the ontologies inconsistent (resp. consistent). The overall benefit of the approach will also be demonstrated by comparing the use of the optimizations previously mentioned with another version of Swoop which does not leverage the techniques.

<sup>1</sup>Swoop Homepage: <http://www.mindswap.org/2004/Swoop/>

<sup>2</sup>Financial OWL Ontology: <http://www.cs.put.poznan.pl/alawrynowicz/financial.owl>

<sup>3</sup>Pellet Homepage: <http://www.mindswap.org/2003/pellet/>