Web Semantics: Science, Services and Agents on the World Wide Web ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Inference Inspector: Improving the verification of ontology authoring actions

Nicolas Matentzoglu [*], Markel Vigo, Caroline Jay, Robert Stevens

School of Computer Science, University of Manchester, Oxford Road, M13 9PL, Manchester, United Kingdom

ARTICLE INFO

Article history:
Received 7 February 2017
Received in revised form 10 June 2017
Accepted 27 September 2017
Available online xxxx

Keywords:
OWL
Ontologies
Human computer interaction
Ontology engineering
Ontology authoring
Reasoning

ABSTRACT

Ontologies are complex systems of axioms in which unanticipated consequences of changes are both frequent, and difficult for ontology authors to apprehend. The effects of modelling actions range from unintended inferences to outright defects such as incoherency or even inconsistency. One of the central ontology authoring activities is verifying that a particular modelling step has had the intended consequences, often with the help of reasoners. For users of Protégé, this involves, for example, exploring the inferred class hierarchy.

This paper provides evidence that making entailment set changes explicit to authors significantly improves the understanding of authoring actions regarding both correctness and speed. This is tested by means of the Inference Inspector, a Protégé plugin we created that provides authors with specific details about the effects of an authoring action. We empirically validate the effectiveness of the Inference Inspector in two studies. In a first, exploratory study we determine the feasibility of the Inference Inspector for supporting verification and isolating authoring actions. In a second, controlled study we formally evaluate the Inference Inspector and determine that making changes to key entailment sets explicit significantly improves author verification compared to the standard static hierarchy/frame-based approach. We discuss the advantages of the Inference Inspector for different types of verification questions and find that our approach is best suited for verifying added restrictions where no new signature, such as class names, is introduced, with a 42% improvement in verification correctness.

1. Introduction

Ontologies are explicit conceptualisations of a domain and are widely applied in biology, health-care and the public domain [1–3]. Ontologies are typically represented in a formal representation language such as the Web Ontology Language (OWL), the Open Biomedical Ontologies format (OBO) or the RDF Schema language (RDFS).[1] The central advantage of using such formalisms is their well-defined semantics. Generic reasoning systems can be used to access knowledge in the ontology that is only implied, i.e. not explicitly stated, allowing richer answers to queries, the identification of inconsistent knowledge and improved management of large terminologies through definition-oriented development. There is strong, if primarily anecdotal, evidence that building ontologies using OWL is difficult and error-prone.

Attempts have been made to quantify this difficulty [5], but there remain many unanswered questions about the cognitive challenge of various ontology authoring tasks such as exploration or modelling. The complexity of OWL [6] can lead to axioms that do not reflect the intentions of the author. Furthermore, the lack of understanding for many of the OWL 2 features (in particular, but not only, by domain experts) can result in unintended inferences, which are often not made explicit by the authoring tool, and even if they are, are rarely communicated to the author clearly. An interview study recently revealed that many ontology experts *frequently* run the reasoner, sometimes *after every modification*, to detect errors such as unsatisfiable classes and to prevent the spread of errors [7]. Participants of that study also felt that the change evaluation phase, i.e. the phase that determines whether a modelling action had the intended consequences, is not well supported by state of the art development tools. Some ontology authors use DL queries, generated on the fly, to do 'spot checks', others work with competency questions that are crafted upfront to automatically verify the correctness of a change. As the conceptual model of an ontology is, however, not always known upfront, competency question-based approaches, perhaps best compared with unit tests in software engineering, are of particular utility later in the engineering process, where their coverage of the ontology depends on the user's diligence. Consequently, we need the user

* Corresponding author.
E-mail addresses: nicolas.matentzoglu@manchester.ac.uk (N. Matentzoglu), markel.vigo@manchester.ac.uk (M. Vigo), caroline.jay@manchester.ac.uk (C. Jay), robert.stevens@manchester.ac.uk (R. Stevens).

[1] OBO and RDFS can be seen as syntactic variations of OWL [4].

https://doi.org/10.1016/j.websem.2017.09.004

interface to reduce the perceived complexity of ontologies, support their evaluation, and help prevent or detect errors.

In this work, we are concerned with improving the evaluation of modelling actions. We call the task of evaluating whether a particular modelling action has had the desired effect "verification". Verification is a key sub-process of ontology authoring that involves conducting a set of tests, for example to make sure that a definition of a class works as intended and that no unsatisfiable classes were introduced [8]. We distinguish in our analysis between different types of verification problems (or questions), which we selected based on our extensive experience with teaching OWL and ontology authoring [9]. Examples of verification problem types are verification of tightened restrictions (for example when adding an existential restriction[2] to a class), verification of fixing an unsatisfiability (did the latest change remove the unsatisfiability?) or verification of a definition (are individuals correctly inferred to be members of that class? Does the class have the expected sub-classes?). An enumeration and definition of such types is necessarily subjective and incomplete; we motivate our selection in the context of the methodological discussion. When developing ontologies with the widely used Protégé ontology engineering environment, the verification step is typically realised by invoking the reasoner and exploring the implicit knowledge in the ontology [7], for example by making sure that a particular class has the expected position in the inferred class hierarchy or a freshly introduced property domain restriction results in the expected individual type inferences. We call this approach *static hierarchy/frame-based* (SHFB), where "static" refers to the fact that the inferred hierarchy only reflects a state, without any indication as to how this state relates to the latest modelling action.

We propose a method to improve verification by presenting changes to finite entailment sets, i.e. sets of axioms of a particular form that are implied by the ontology [10]. A prominent example of a finite entailment set is the set of all implied subclass (or disjointness) relations between classes in the ontology. We explore the hypothesis that making changes to key entailment sets explicit improves verification compared to the static hierarchy/frame-based approach.

Our contributions are as follows:

- We developed the Inference Inspector, a novel Protégé plugin that alerts the author to the changes to key entailment sets that have occurred as a consequence of a modelling action.
- We conducted an exploratory study to evaluate our Inference Inspector prototype. We find that our approach is better suited to tasks that involve changing definitions or adding restrictions on existing entities, and less well suited for tasks that involve the introduction of new entities, compared to the SHFB approach.
- We conducted a laboratory experiment that confirms our hypothesis. We find that making entailment set changes explicit improves the understanding of consequences both in terms of correctness and speed, and is rated as the preferred way to inspect the consequences of changes, compared to the SHFB approach.

2. Background and related work

Ontology authoring is the creation and maintenance of ontology artefacts constructed using a formal knowledge representation language such as OWL, OBO or RDFS. We view an ontology $\mathcal{O}$ as a set of axioms, with $\alpha \in \mathcal{O}$ being an axiom in $\mathcal{O}$. The signature of $\mathcal{O}$, denoted $\widetilde{\mathcal{O}}$, is the set of logically relevant entities, i.e. classes, object properties, data properties and individuals,[3] across all axioms in $\mathcal{O}$. Given a language $\mathcal{L}$ and an OWL 2 ontology $\mathcal{O}$, the $\mathcal{L}$-entailment set of $\mathcal{O}$, written $\mathcal{E}(\mathcal{O}, \mathcal{L})$, is the set of all axioms in $\mathcal{L}$ that are entailed by $\mathcal{O}$ (entailment set). One of the most important entailment sets for ontology authoring is the set of atomic subsumptions, i.e. the language $\mathcal{L}$ that allows us to build axioms of the form *SubClassOf*(A, B) for all valid combinations of A and B in signature $\widetilde{\mathcal{O}} \cup \top \cup \bot$. More precise definitions of the entailment sets relevant to this work can be found in Appendix A.

Typical ontology authoring activities include, but are not limited to, the creation of axioms or annotations. For a detailed discussion of ontology authoring activities see [11]. While ontology authoring is increasingly performed in a programmatic fashion, a large number of ontologies have been built using ontology authoring environments such as Protégé [12] and WebProtégé [13]. Moreover, based on our experience, even if ontologies are created in a programmatic fashion, they are often checked for defects in a visual authoring environment.

Research on ontology authoring has experienced a resurgence in recent years [7,11,14], due at least in part to the increased availability of change-logs for ontology development. WebProtégé, for example, produces detailed change-logs, which can form the basis of a rich and informative analysis of ontology authoring activities [14]. The work presented here builds on a series of investigations into the ontology authoring process [7,11,15]. The aim of this body of work is to improve our understanding of ontology authoring processes, and in particular to identify typical authoring styles and workflows that will help tool developers to improve their support of the authoring process. We have identified a number of difficulties shared across ontology developers that occur during ontology development, and have found that Protégé does not support the needs of current authors [7,15]. In particular, ever more sophisticated ontology modelling patterns make the verification of modelling actions difficult. For example, the combination of a *SubClassOf* axiom such as *SubClassOf*$(A, R$ some $B)$ with a *ObjectPropertyDomain*(R, C) can influence the class hierarchy, at least for a significant proportion of the OWL 2 users, unexpectedly.[4] The fact that unintended consequences such as the introduction of unsatisfiable classes, broken definitions (that result in inaccurate classifications) or inaccurate inferences on the data level (ABox) are often difficult to spot was one of the core incentives for this work. A specially modified version of Protégé that collects interaction events silently during ontology authoring [11], Protégé4US, enabled us to study ontology authoring workflows and derive a number of well-founded design suggestions for authoring tools [11]. One of these was *making the changes to the inferred hierarchy explicit*— another major incentive for developing the Inference Inspector.

Developing better support for ontology authoring is a long-standing challenge spanning a number of domains including visualisation [16] and debugging [17]. Ontology visualisations, concerned with developing (scalable) visualisations of ontology concepts, their instances and inter-relations, support ontology authors in exploration tasks (such as verification or familiarisation) and making portions of the ontology (such as justifications) easier to understand, for example using graphs or graph-like structures, natural language [18], diagrammatic representations, such as crop-circles [19] or, for simpler ontologies, graphs [20]. While the majority of ontology visualisation focuses on graph-like representations (2D and 3D) and tree-like hierarchical structures [16], efforts are

[2] I.e. adding an axiom of the form *SubClassOf*$(A, R$ some $B)$ for a class A.

[3] For our purposes, this excludes in particular annotation properties and datatypes, which are considered entities in OWL 2, but are not of interest to logical reasoning.

[4] Namely, by causing A to be a subclass of C.