Explaining justifications in OWL DL ontologies

Conference Item


For guidance on citations see FAQs

© 2011 The Open University
Version: Version of Record

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online’s data policy on reuse of materials please consult the policies page.

oro.open.ac.uk
Explaining Justifications in OWL DL Ontologies

Tu Anh T. Nguyen
Department of Computing
The Open University, UK
t.nguyen@open.ac.uk

1 Research Problem

Building domain ontologies such as the SNOMED CT\textsuperscript{1} requires a significant amount of effort and expertise. Among supporting tools for building such ontologies are tableaux-based reasoners \cite{21,19} which can help with detecting inconsistencies or computing implicit statements (i.e. entailments) following from ontologies. They do, however, have a downside: it is difficult to explain their reasoning to human ontology developers as they depend on a refutation proof strategy and unintuitive normalising transformations \cite{2}. Therefore, we aim to develop a computational approach to generating explanations for inconsistencies and undesirable entailments in order to support ontology debugging.

Recently there has been an increasing interest in justifications for entailments and inconsistencies \cite{11,20,7}. A justification for an entailment is a minimal set of axioms, drawn from the ontology, that causes the entailment to arise. For logicians, a justification is a form of explanation \cite{11}. Human domain experts do, however, require further explanations of the semantics of each axiom and the inference process linking axioms to the entailment \cite{18}. This is because they usually lack understanding of OWL and formal logic \cite{4}.

Let us give an example. Assuming a justification for an unwanted entailment $\text{Animal} \equiv \top$ includes three axioms: (1) $\text{Old Lady} \equiv \forall \text{has pet}. \text{Cat}$, (2) $\text{Domain(\text{has pet})} = \text{Animal}$, and (3) $\text{Old Lady} \sqsubseteq \text{Animal}$. We may provide an explanation as the following\textsuperscript{2}:

Everything is an $\text{Animal}$ as (a) everything that has no pets at all is an $\text{Animal}$, and (b) everything that has at least one pet is an $\text{Animal}$.

Statement (a) is inferred from axioms 1 and 3. From axiom 1, we know that (c) everything that has no pets at all is an $\text{Old Lady}$, and from axiom 3, we know that (d) every $\text{Old Lady}$ is an $\text{Animal}$. Statement (c) is caused by the semantics of the universal quantifier; that is, it accepts everything that has no $\text{has pet}$ relations. Statement (b) is inferred from axiom 2.

We observe that automatically generated justifications and recent research on mapping between OWL and controlled English \cite{17} do provide a basis for constructing explanations such as the one above. The expected results of our research may be applied as a debugging support in ontology editors.

\textsuperscript{1} See http://www.ihtsdo.org/snomed-ct/

\textsuperscript{2} This explanation is aimed as an attempt, not our final output.
2 Related Work

McGuinness first investigated the problem of explaining entailments generated by a normalize-compare reasoner, whose operation was highly optimized but unintuitive to humans [13]. She proposed a proof-theoretic framework for constructing proofs as explanations, using inference rules. However, the framework was unable to deal with formalisms more sophisticated than the “core description logic” [15], and provided only logical proofs but not explanations for end-users.

Borgida and colleagues proposed another framework for constructing proofs for subsumption entailments computed by tableaux-based reasoners [2]. They constructed proofs which included steps that parallel major steps in the tableaux algorithm. These steps were then linked by using inference rules. Based on Borgida’s work, Kwong implemented a tool that generated explanations for ALC subsumptions [12]. However, the style and structure of such explanations served for proof checking rather than proof understanding for end-users [8].

Regarding the use of justifications, a framework for constructing “justification oriented proofs” has recently been proposed by Horridge [9]. In this framework, candidate lemmas of some fixed forms are collected from the deductive closure of the input justification. Based on these lemmas, all possible proofs are constructed, and the one whose “structural complexity” is minimal will be selected. We believe that proofs constructed based on their structural complexity may not provide good understanding of the reasoning for end-users; and that research from the other end of explanations such as human deductive reasoning ability and what people want to get from a justification would provide good suggestions on producing explanations accessible for humans. Moreover, many unwanted entailments are caused by the misunderstanding of ontology developers on the semantics of OWL functors [3]; thus, some kind of explanations would be more helpful than showing logical proofs.

3 Proposed Approach and Methodology

We use justifications as inputs for generating explanations for entailments and inconsistencies. Since OWL-DL justifications can be as expressive and diverse as OWL-DL ontologies, it is very difficult to cover all possible justifications. Moreover, in practice human ontology developers overwhelmingly favour relatively simple OWL functors [17] and ontology design patterns\(^3\). Therefore, we narrow our research to explaining justifications that are commonly used and efficiently used in ontology engineering. Our proposed approach consists of three parallel and overlapping stages as shown in Figure 1.

The first stage is to collect a comprehensive corpus of justification patterns used in ontology engineering. We will collect justifications from existing OWL-DL ontologies, especially key domain ontologies such as SNOMED CT. Justifications are then categorised into patterns at different levels of abstraction. The

\(^3\) See \url{http://ontologydesignpatterns.org/wiki/Main_Page}
most specific pattern will rely on justifications’ structure. It retains all OWL
functors, but abstracts over atomic terms; this is done by substituting names
of entities (i.e. classes, individuals, properties, . . . ) by alpha-numeric identifiers.
For example, names of atomic classes are replaced by capital letters $A$, $B$, . . . ,
those of object properties by $r0$, $r1$, . . . Based on specific patterns, abstract pat-
terns will be formulated by abstracting over groups of axioms which have certain
inference steps in common. An example specific pattern and its abstract pattern
are shown in Table 1. Proper studies on the usefulness and completeness of our
pattern corpus will also be investigated.

In the second stage, we explore ways of structuring and expressing expla-
nations for our patterns by looking at the user end of explanations. We will
look at psychology aspects of human deductive reasoning [16] and analyse users’
expectations when reading justifications. We will also collect or manually work
out explanations for each pattern, then evaluate them through user studies. In
this way, we can collect a corpus of good human-generated explanations for our
patterns as suggestions on structuring machine-generated explanations.

The last stage is to automate the generation of explanations. Based on sug-
gestions collected from the previous stage, we will produce discourse plans for
our patterns and verbalise them in English. Enhanced techniques for planning
output texts [10, 5] and recent results in mapping between OWL and controlled
English [17] will be considered to apply.

Regarding evaluations, human judgement about the readability and under-
standability of explanations plays an important part. We plan to evaluate both
discourse plans and explanations. Discourse plans are tested with logicians and
explanations are tested with non-expert ontology developers. In both studies,
participants will be asked to carry out a comprehensive task to see whether they
perform better after reading our explanations than reading justifications. Participants’
feedbacks on the style and structure of explanations will also be collected.
Another method of evaluation is to collect and compare our explanations with
those of other explaining systems.

4 Results

We have been working on the first two stages of our approach. Particularly,
we developed a Java-based program to collect and classify justifications into
specific patterns based on their structure, relying on the Pellet and FaCT++ and the program of Horridge [6] for computing regular and laconic justifications for entailments. An empirical study into commonly used justification patterns from 191 ontologies of varying size, subject and expressivity of the TONES Ontology Repository [1] was carried out. Justifications having common specific patterns were grouped, and their frequencies were calculated by two different measures: (a) occurrences of the justification pattern across all ontologies (called ‘justification frequency’), and (b) the number of ontologies in which the pattern occurred at least once (called ‘ontology frequency’). Our results show that some specific patterns, often relatively simple, are used more frequently than others, and that it is possible to represent most specific patterns by a relatively small number of abstract patterns [14].

Table 1 shows a specific pattern and its abstract pattern we collected. \(\text{CLASS-INCLUSION-CHAIN}(B,A)\) represents a chain of class inclusion axioms (i.e. \(\text{SubClassOf}\) and \(\text{EquivalentClass}\)) from named class \(B\) to named class \(A\). The ontology and justification frequencies of the specific pattern are 8.9% and 1.5%; and those of the abstract one are 22.6% and 27.8% respectively.

Table 1. A specific pattern and its abstract pattern from the TONES Repository

<table>
<thead>
<tr>
<th>Specific Pattern</th>
<th>Abstract Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entailment (A \equiv \top)</td>
<td>Entailment (A \equiv \top)</td>
</tr>
<tr>
<td>Justification (+ B \equiv Vr0.C)</td>
<td>Justification (+ B \equiv Vr0.C)</td>
</tr>
<tr>
<td>(+ \text{Domain}(r0) = A)</td>
<td>(+ \text{Domain}(r0) = A)</td>
</tr>
<tr>
<td>(+ B \subseteq D)</td>
<td>(+ \text{CLASS-INCLUSION-CHAIN}(B,A))</td>
</tr>
<tr>
<td>(+ D = A)</td>
<td></td>
</tr>
</tbody>
</table>

5 Conclusions and Future Work

We have presented an outline of our research that uses justifications as a basis for providing natural language explanations for unwanted entailments and inconsistencies in OWL-DL ontologies. The expected results of our research might be applied as a debugging support for end-users, especially human domain experts, while they develop ontologies. We have completed the classification of justifications into patterns and an empirical study on commonly used patterns. The next steps are to continue working out our corpus of justification patterns, to explore ways of structuring and expressing explanations, to develop methods for producing discourse plans and realising them in English, and finally, to evaluate our results.

Acknowledgments. The research described in this paper was undertaken as part of the ongoing SWAT project (Semantic Web Authoring Tool), which is supported by the UK Engineering and Physical Sciences Research Council (EPSRC). We thank our colleagues and the anonymous viewers.
References