From Justifications Towards Proofs For Ontology Engineering

Matthew Horridge and **Bijan Parsia**
School of Computer Science
University of Manchester, M13 9PL UK

Introduction

Over the past few years there has been a significant amount of interest in the area of explaining entailments in OWL ontologies[1]. Without some kind of tool support, it can be very difficult, or even impossible, to work out why entailments arise in ontologies. Even in small ontologies that only contain tens of axioms, there can be multiple reasons for an entailment, none of which may be obvious. It is for this reason that there has recently been a lot of focus on generating explanations for entailments in ontologies. In the OWL world, *justifications* are a popular form of explanation for entailments. A justification is a minimal subset of an ontology that is sufficient for an entailment to hold (Baader and Hollunder 1995; Schlobach and Cornet 2003; Kalyanpur 2006). More precisely, for an ontology $\mathcal{O}$ and an entailment η where $\mathcal{O} \models \eta$ ($\mathcal{O}$ entails η), a set of axioms $\mathcal{J}$ is a justification for η with respect to $\mathcal{O}$ if $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \eta$ and, for all $\mathcal{J}' \subsetneq \mathcal{J}$, $\mathcal{J}' \not\models \eta$. Additionally, $\mathcal{J}$ is simply a justification (without reference to $\mathcal{O}$) if $\mathcal{J} \models \eta$ and, if $\mathcal{J}' \subsetneq \mathcal{J}$, then $\mathcal{J}' \not\models \eta$.

Despite the utility of justifications, in certain cases people can struggle to understand how a justification supports an entailment. Indeed, in a user study detailed in (Horridge, Parsia, and Sattler 2009), it was found that a wide range of people, who have expertise in building OWL ontologies, can find it very difficult and even impossible to understand *some* justifications for entailments in *real* ontologies. An example justification that many people found difficult to understand is shown in Figure 1, which is for the entailment $\mathsf{Person} \sqsubseteq \bot$ (i.e. Person is unsatisfiable). In this justification, Movie is entailed to be equivalent to $\top$ but many people failed to spot this. Two important results emerged from the study: 1) There were large numbers of justifications that *all* participants *could* understand (including non-trivial justifications containing general concept inclusions, transitivity etc.). This includes participants with limited experience of OWL and description logic, and people who had never encountered justifications before. This suggests that justifications have merit as a form of explanation. 2) There were

$\mathsf{Person} \sqsubseteq \neg\mathsf{Movie}$
$\mathsf{RRated} \sqsubseteq \mathsf{CatMovie}$
$\mathsf{CatMovie} \sqsubseteq \mathsf{Movie}$
$\mathsf{RRated} \equiv (\exists \mathsf{hasScript.TS}) \sqcup (\forall \mathsf{hasViolenceLevel.High})$
$\mathsf{Domain}(\mathsf{hasViolenceLevel},\ \mathsf{Movie})$

Figure 1: A justification for $\mathsf{Person} \sqsubseteq \bot$

a number of justifications that *all* participants ranked “difficult” to “impossible” to understand. This includes people who have over two years experience of working with OWL, building ontologies and even includes people who have developed OWL reasoners. This is indicative that justification understanding can be a real problem.

The work presented in this paper looks at a possible solution to the problem of understanding justifications. A framework is presented which is used to *lemmatise justifications*. The ultimate goal is that this lemmatisation process could be used to produce *justification oriented proofs*, which show, in a stepwise way using justifications how the lemmas, and ultimately the entailment, follows from a justification. The work presented in this paper is applicable to OWL 2, which is underpinned by the $\mathcal{SROIQ}$ description logic, but could be applied to all First Order Logic Fragments.[2]

Justification Oriented Proofs

The main idea behind a justification oriented proof is depicted in Figure 2. The numbered rectangles represent axioms, with the rightmost rectangle, labelled η, representing the entailment of interest. The shaded rectangles labelled with “1” – “6” represent exactly the axioms that appear in the original justification $\mathcal{J}$ for the entailment (and are therefore in the ontology as asserted axioms). Axioms “7” and “8” are lemmas. Axioms “1”, “2” and “3” are a justification for axiom “7” and axioms “3”, “4” and “5” are a justification for axiom “8”. Together axioms “6”, “7” and “8” constitute a justification for η i.e. the entailment. In essence, the process of understanding why a justification supports an entailment, is transformed to understanding how subsets of the justification result in intermediate entailments, and understanding how these intermediate entailments fit together to give rise the main entailment of interest.

[1]An OWL (more precisely OWL 2) ontology may be regarded as a $\mathcal{SROIQ}$ knowledge base

[2]A longer version of this paper may be found at `http://owl.cs.manchester.ac.uk/explanation/lemmas`

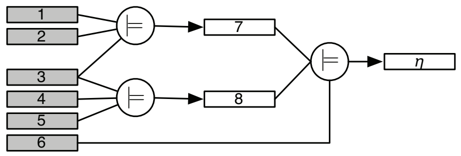

Figure 2: A schematic of a Justification Oriented Proof

Justification Lemmatisation

Given a justification $\mathcal{J}$ for an entailment η, $\mathcal{J}$ can be lemmatised into $\mathcal{J}'$, so that $\mathcal{J}'$ is *easier to understand than* $\mathcal{J}$. With this notion in hand, lemmas for justifications can now be defined. First, an informal description is given, then a more precise definition is given in Definition 2. Informally, a set of lemmas $\Lambda_{\mathcal{S}}$ for a justification $\mathcal{J}$ for η is a set of axioms that is entailed by $\mathcal{J}$ which can be used to replace some set $\mathcal{S} \subseteq \mathcal{J}$ to give a new justification $\mathcal{J}' = (\mathcal{J} \setminus \mathcal{S}) \cup \Lambda_{\mathcal{S}}$ for η. Moreover, $\mathcal{J}'$ is simpler to understand than $\mathcal{J}$. $\mathcal{J}'$ is called a *lemmatisation of* $\mathcal{J}$. Various restrictions are placed on the generation of the set of lemmas $\Lambda_{\mathcal{S}}$ that can lemmatise a justification $\mathcal{J}$. These restrictions prevent "trivial" lemmatisations, an example of which will be given below. Before these restrictions are discussed, it is useful to introduce the notion of a *tidy* set of axioms. Intuitively, a set of axioms is *tidy* if it is consistent, contains no synonyms of $\bot$ (where a class name is a synonym of $\bot$ with respect to a set of axioms $\mathcal{S}$ if $\mathcal{S} \models A \sqsubseteq \bot$), and contains no synonyms of $\top$ (where a class name is a synonym of $\top$ with respect to a set of axioms $\mathcal{S}$ if $\mathcal{S} \models \top \sqsubseteq A$).

Definition 1 (Tidy sets of axioms) *A set of axioms $\mathcal{S}$ is* tidy *if $\mathcal{S} \not\models \top \sqsubseteq \bot$, $\mathcal{S} \not\models A \sqsubseteq \bot$ for all $A \in Signature(\mathcal{S})$, and $\mathcal{S} \not\models \top \sqsubseteq A$ for all $A \in Signature(\mathcal{S})$.*

The restrictions mandate that a set of lemmas $\Lambda_{\mathcal{S}}$ must only be drawn from (i) the deductive closure of *tidy* subsets of the set $\mathcal{S} \subseteq \mathcal{J}$, (ii) from the *exact* set of synonyms of $\bot$ or $\top$ over $\mathcal{S}$. Without the above restrictions on the axioms in $\Lambda_{\mathcal{S}}$, it would be possible to lemmatise a justification $\mathcal{J}$ to produce a justification $\mathcal{J}'$ that, in isolation, is simple to understand, but otherwise bears little or no resemblance to $\mathcal{J}$. For example, consider $\mathcal{J} = \{A \sqsubseteq \exists R.B,\ B \sqsubseteq E \sqcap \exists S.C,\ B \sqsubseteq D \sqcap \forall S.\neg C\}$ as a justification for $A \sqsubseteq \bot$. Suppose that *any* axioms entailed by $\mathcal{J}$, could be used as lemmas (i.e. there are no restrictions on the axioms that make up $\Lambda_{\mathcal{S}}$). In this example, A is unsatisfiable in $\mathcal{J}$, meaning that it would be possible for $\mathcal{J}' = \{A \sqsubseteq E, A \sqsubseteq \neg E\}$ to be a lemmatisation of $\mathcal{J}$. Here, $\mathcal{J}'$ is arguably easier to understand than $\mathcal{J}$, but in bears little resemblance to $\mathcal{J}$. In other words, $A \sqsubseteq E$ and $A \sqsubseteq \neg E$ are not helpful lemmas for $\mathcal{J} \models A \sqsubseteq \bot$. Similarly, unhelpful results arise if lemmas are drawn from *inconsistent* sets of axioms, or sets of axioms that contain synonyms for $\top$.

Lemmas for Justifications Defined

In what follows, δ is the 'well known' structural transformation originally defined in (Plaisted and Greenbaum 1986). This structural transformation pulls axioms apart and flattens out concept expressions, removing any nesting, and is used in order to allow a "fine-grained" approach in lemma generation. In what follows $\mathcal{T}^\star$ is the deductive closure of $\mathcal{T}$, $\mathcal{J}^\star$ is the deductive closure of $\mathcal{J}$, A represents an atomic class name, and $Complexity$ is a function that returns a value that represents how complex a justification is *for some purpose*—the larger the value the more complex. In essence the complexity function is used to choose one lemmatised justification over another.

Definition 2 (Lemmas for Justifications) *Let $\mathcal{J}$ be a justification for η and $\mathcal{S}$ a set of axioms such that $\mathcal{S} \subseteq \mathcal{J}$. Let $\Theta_{\mathcal{S}}$ be the set of tidy subsets of $(\mathcal{S} \cup \delta(\mathcal{S}))$. Let $\Omega_{\mathcal{S}}$ be the set of consistent subsets of $(\mathcal{S} \cup \delta(\mathcal{S}))$. Let*

$$\Lambda_{\mathcal{S}} \subseteq \bigcup_{\mathcal{T} \in \Theta_{\mathcal{S}}} \mathcal{T}^\star \cup \{\alpha \mid \alpha \textit{ is of the form } A \sqsubseteq \bot \textit{ or } \top \sqsubseteq A, \textit{ and } \exists \mathcal{K} \in \Omega_{\mathcal{S}} \textit{ s.t. } \mathcal{K} \models \alpha\}$$

$\Lambda_{\mathcal{S}}$ is a set of lemmas *for a justification $\mathcal{J}$ for η if, for $\mathcal{J}' = (\mathcal{J} \setminus \mathcal{S}) \cup \Lambda_{\mathcal{S}}$*

1. *$\mathcal{J}'$ is a justification for η over $\mathcal{J}^\star$, and,*
2. *$Complexity(\eta, \mathcal{J}') < Complexity(\eta, \mathcal{J})$.*

Complexity Models

As can be seen, Definition 2 relies on a function that provides a complexity score for a justification. The higher the score the more complex the justification is for some particular purpose. In terms of user understanding, a hard to understand justification has a higher score than an easy to understand justification. It should be noted that the framework presented here is intended to be rather general. The complexity function should be thought of as being "pluggable". In the work presented here, a simple model for predicting how complex a justification is for a person to understand is used. This model could of course be adapted and tailored for specific audiences. It is easy to imagine that a tool that provides explanations to end users could have various strengths of lemmatisation that correspond to different complexity models under the hood. Briefly, the model is composed of two *main* parts: 1) A "structural" complexity measure, which estimates the complexity of a justification based on metrics such as the number of *different types* of axioms and *different types* of concept expressions that appear in a justification, how the signature is distributed over axioms etc.; 2) A "specific phenomena" based complexity measure, which increases the complexity of a justification when certain problematic patterns of axioms or kinds of problematic entailments, identified in the user study, occur in the justification.

An Example

An example of the kinds of lemmas and lemmatised justifications that get computed for the justification shown in Figure 1 is presented as a justification oriented proof in Figure 3. Note that the presentation style used here is merely for illustrative purposes, rather than as an end user presentation device. The axioms shown in bold are the axioms that appear in the original justification, and are therefore asserted, and all other axioms correspond to lemmas. Notice that axioms at each level of indentation form a justification for the

- $\mathrm{Entailment} : \mathsf{Person} \sqsubseteq \bot$ (1)
- $\mathbf{Person} \sqsubseteq \neg\mathbf{Movie}$ (2)
- $\top \sqsubseteq \mathsf{Movie}$ (3)
  - $\forall\mathsf{hasViolenceLevel}.\bot \sqsubseteq \mathsf{Movie}$ (4)
    - $\forall\mathsf{hasViolenceLevel}.\bot \sqsubseteq \mathsf{RRated}$ (5)
      - $\mathbf{RRated} \equiv (\exists\mathbf{hasScript}.\mathbf{TS}) \sqcup (\forall\mathbf{hasViolenceLevel}.\mathbf{High})$ (6)
    - $\mathsf{RRated} \sqsubseteq \mathsf{Movie}$ (7)
      - $\mathbf{RRated} \sqsubseteq \mathbf{CatMovie}$ (8)
      - $\mathbf{CatMovie} \sqsubseteq \mathbf{Movie}$ (9)
  - $\exists\mathsf{hasViolenceLevel}.\top \sqsubseteq \mathsf{Movie}$ (10)
    - $\mathbf{Domain}(\mathbf{hasViolenceLevel}, \mathbf{Movie})$ (11)

Figure 3: Example Lemmatisations

axiom that is above them. The example illustrates how a justification oriented proof could eventually be obtained by first computing a lemmatised justification for the original entailment and the recursively computing justifications for each lemma and lemmatising these as necessary.

In the presentation here, the example shown in Figure 3 may be read as follows. Person is unsatisfiable because Person is disjoint with Movie (axiom 1, asserted, and thus bold) and yet everything must be a Movie (axiom 2, a lemma, generated by our approach). The level below $\top \sqsubseteq$ Movie, corresponding to axiom 3 and axiom 9, explains why everything must be a Movie. This is due to the fact that everything that does not have a violence level is a Movie (axiom 3), and everything that does have a violence level is a Movie (axiom 9). The reason that anything that does not have a violence level is a Movie is due to axioms 4 and 6, both of which are lemmas, which form a justification saying that anything that does not have a violence level is RRated and anything that is RRated is a Movie. Axiom 4, which specifies that anything that does not have a violence level is RRated, is a lemma which is entailed by one asserted axiom, i.e. axiom 5. Similarly, the lemma corresponding to axiom 6 is entailed by two asserted axioms, 7 and 8. Finally, the lemma that everything that has a violence level is a Movie is entailed by axiom 10, the asserted domain axiom.

While the presentation in Figure 3 is for illustrative purposes only, one could imagine an interactive debugging and explanation tool that uses the underlying proof structures to derive a user friendly proof presentation. Such a presentation may offer the ability to expand and collapse various levels. Additionally, the presentation used here is a "top down" presentation which goes from entailment to asserted axioms. The alternative presentation would be a "bottom up" presentation that would go from asserted axioms, via lemmas, to the entailment.

Considerations

Labelling Lemmatisations with Rules In the work by Borgida et al.(Borgida, Franconi, and Horrocks 2000), and other Natural Deduction inspired proofs each step in an explanation is labelled with the name of a rule. The rule name identifies the syntactic transformation that is performed in the step. The question of whether it is useful to label steps in a proof therefore arises. The main advantage of such a labelling is that it could provide a link to a catalogue of explanation "rules" or common situations, that explain the rationale behind the rule. However, the fact that participants in the user study could understand many justifications indicates that rule labelling is not *always* necessary.

Granularity of Resulting Proofs It is easy to imagine that when used to construct a justification oriented proof an overly aggressive lemmatisation could introduce too many steps. In these cases the lemmatisation process would have a detrimental effect on end user understanding. The chances of this happening could be minimised by careful construction of the complexity model so that a justification is only lemmatised when necessary. An alternative solution is to design end user interaction mechanisms that would support an initially conservative approach to lemmatisation with it being possible to request further lemmatisation where necessary.

Conclusions and Future Work

Some naturally occurring justifications can be difficult for a range people to understand. However, there are justifications that a range of people *can understand*, including people who have never worked with justifications before. In the approach presented here justification lemmatisation is used to draw out intermediate conclusions that need to be spotted in order for people to understand a justification. Given a justification for an entailment, a lemmatisation produces a new justification for the entailment that is less complex for some purpose (for example end user understanding). The framework presented in this paper is intended to be rather general.

Finally, it is envisaged that lemmatised justifications could be stitched together into justification oriented proofs, which break justifications down into series of smaller easier to understand justifications. While a general framework for justification lemmatisation has been presented and investigated here, more work is needed to investigate the kinds of complexity models and candidate lemmatisation routines that give rise to good proofs. Moreover, presentation mechanisms for justification oriented proofs need to be devised so that it is possible to investigate and confirm that justification oriented proofs are indeed beneficial to end users.

References

Baader, F., and Hollunder, B. 1995. Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning* 14(1):149–180.

Borgida, A.; Franconi, E.; and Horrocks, I. 2000. Explaining $\mathcal{ALC}$ subsumption. In *ECAI 2000*.

Horridge, M.; Parsia, B.; and Sattler, U. 2009. Lemmas for justifications in OWL. In *DL 2009*.

Kalyanpur, A. 2006. *Debugging and Repair of OWL Ontologies*. Ph.D. Dissertation, U. Maryland.

Plaisted, D. A., and Greenbaum, S. 1986. A structure-preserving clause form translation. *Journal of Symbolic Computation*.

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