From Justifications Towards Proofs For Ontology Engineering

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

Abstract

Justifications — that is, minimal entailing subsets — are the current dominant form of explanation service provided by ontology engineering environments, especially those focused on the Web Ontology Language (OWL). This is a somewhat surprising development: The standard candidate for explanations of entailments are *proofs*, and this was so in ontology engineering prior to 2003. Justifications are merely the *premises* of a proof and, as such, do not articulate the (often non-obvious) reasoning which connect those premises with the conclusion.

In this paper we argue that justifications are sufficient for many ontology engineering purposes and why proofs may be, in general, counterproductive. However, we also provide evidence that certain proof like notions might be helpful in a justification oriented environment.

Introduction

The Web Ontology Language (OWL) is a W3C standard and has become one of the most prominent, if not the most[1] prominent, ontology language. With an expanding base of users, both academic and industrial, the need for good ontology engineering services has become more pressing.

OWL[2] is based on the Description Logic (DL) $\mathcal{SROIQ}(D)$, and thus falls into the family of first order logic based KR languages.

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; Kalyanpur 2006; Schlobach and Cornet 2003). 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 respect to $\mathcal{O}$) if $\mathcal{J} \models \eta$ and, if $\mathcal{J}' \subsetneq \mathcal{J}$, then $\mathcal{J}' \not\models \eta$.

Background

In determine the sorts of explanations we should provide (or investigate providing) or whether a certain form of explantion is adequate, we need to consider at least three factors:

1. The characteristics of the reasonably typical ontology, explanandum, and available explicans. For example, if we consider the explanation of subsumptions in RDFS knowledge bases which only have rdfs:subClassOf assertions, every possible explicans is going to involve "pointer chasing" along explicit subclass paths. In this case, we might dispense with any textual or specific explandums in favor of a visualization of the subsumption hierarchy.
2. The characteristics of the targeted tasks both fine grained and general. For example, there's a typically considerable difference in the sorts of explanations needed for ontology development and for end user deployment. Similarly, *debugging* unsatisfiable classes (where modification of the ontology is a goal) is quite different from sanity checking a general entailment, or just for getting a sense of the ontology.

   This is even clearer in pedagogic contexts: For example, the kinds of proofs you ought to supply will be determined by the type of proof procedure you are trying to teach!
3. The characteristics of your target audience. For example, we might target novice or experienced ontology developers, or, orthogonally, ontology developers who are primarily domain experts and ontology developers who are primarily knowledge engineers. Target audiences vary greatly in their background knowledge, willingness to learn new things, and willingness to tolerate additional complexity.

[1] How, exactly, to determine this is quite difficult both because the data is hard to collect and because there are many dimensions of "prominence". For example, while the Protégé home page(**?**) lists 3,293 members of the `protege-discussion` mailing list (which discusses the Protégé-Frames tool and langauge) while only 2,254 for the `protege-owl` list, the mailing lists page sets expectations of 5-10 messages/day for the former and 15-20 for the latter. Regardless, it's clear that OWL is as mass market an ontology language as one can hope for.

[2] In this paper, we will unless otherwise indicated, use "OWL" to refer to the OWL 2 DL flavour of OWL.

To illustrate the effects of these factors, consider the following scenarios:

- **Explanation of (novel) mathematical theorems found by an automated reasoner.** In this case, the formalism is typically going to be at least full first order logic and the "ontology" is typically going to be a fairly small set of axioms which are highly trusted (think, Euclid's axioms, or Peano's axioms). That is, the user is interested primarily in whether the given axioms support the given theorem. The typical task in this scenario is proving the theorem follows from the axiom and the audience is very mathematically sophisticated and while strongly interested in *whether* the axioms support the theorem are also strongly interested in *how*. Proofs are key working objects for htem. Also, their "how" interest is often less in the formal details than in the proof strategy.
- **Explanation in a clinical support system.** Knowledge bases for diagnosis in expert systems such as Mycin have classically consisted of a set of *rules* where the rules, from a formalism perspective, are production rules (which may more or less correspond to a logic programming like logic), but from a modeller and user perspective correspond to many different sorts of rules, including rules of thumb. The explanadum (a diagnosis or treatment) has a corresponding decision tree.

  In this case, the knowledge base is a large collection of rules connecting evidence to diagnostic judgments in a way that largely mirrors the decision trees of the users. The users generally presume that the *modelling* is correct, but they typically need more information than the raw dignosis in order to determine proper care.
- **Explanation of an unsatisfiable class in an OWL ontology.**

Justifications

Definition 1 (Justification) *$\mathcal{J}$ is a justification for $\mathcal{O} \models \eta$ 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 respect to $\mathcal{O}$) if $\mathcal{J} \models \eta$ and, if $\mathcal{J}' \subsetneq \mathcal{J}$, then $\mathcal{J}' \not\models \eta$.*

Example An issue with justification is that they operate at the level of *asserted* axioms. That is, a justification can only contain axioms that are contained within an ontology. Because OWL allows the nesting of complex class expressions, and tools make it easy to construct ontologies that contain "long" axioms, it is frequently the case that, while *all axioms* are required for an entailment to hold, not *all parts* of axioms within a justification are required for an entailment to hold.

Example

Laconic Justifications

Intuitively, a justification $\mathcal{J}$ for an entailment η in an ontology $\mathcal{O}$ is a *laconic justification* if all of its axioms do not contain any superfluous parts and moreover all parts of axioms are as weak as possible (Horridge, Parsia, and Sattler 2008).

In order to define laconic justifications more precisely, several pieces of machinery need to be introduced. The first is the "well known" satisfiability preserving *structural transformation* δ, which flattens out axioms, removing any nesting of complex class expressions. The structural transformation was first defined in (Plaisted and Greenbaum 1986), with a version of the rewrite rules for description logic syntax given in (Motik, Shearer, and Horrocks 2007).

The Structural Transformation δ

$$\delta(\mathcal{O}) := \bigcup_{\alpha \in \mathcal{R} \cup \mathcal{A}} \delta(\alpha) \cup \bigcup_{C_1 \sqsubseteq C_2 \in \mathcal{T}} \delta(\top \sqsubseteq \mathsf{nnf}(\neg C_1 \sqcup C_2))$$
$$\delta(D(a)) := \delta(\top \sqsubseteq \neg\{a\} \sqcup \mathsf{nnf}(D))$$
$$\delta(\top \sqsubseteq \mathbf{C} \sqcup \exists R.D) := \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq \exists R.A'_D\} \cup \delta(A'_D \sqsubseteq D)$$
$$\delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.D) := \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq \forall R.A'_D\} \cup \delta(A'_D \sqsubseteq D)$$
$$\delta(\top \sqsubseteq \mathbf{C} \sqcup \geq nR.D) := \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq \geq nR.A'_D\} \cup \delta(A'_D \sqsubseteq D)$$
$$\delta(\top \sqsubseteq \mathbf{C} \sqcup \leq R.D) := \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq \leq nR.A'_D\} \cup \delta(A'_D \sqsubseteq D)$$
$$\delta(A'_D \sqsubseteq D) := \delta(A'_D \sqsubseteq D) \text{ (If } D \text{ is of the form } A \text{ or } \neg A)$$
$$\delta(A'_D \sqsubseteq D) := \delta(\top \sqsubseteq \neg A'_D \sqcup D) \text{ (If } D \text{ is } \textit{not} \text{ of the form } A \text{ or } \neg A)$$
$$\delta(\beta) := \beta \text{ for any other axiom}$$

Note: A is an atomic concept in the signature of $\mathcal{O}$, A_D and A'_D are fresh concept names that are not in the signature of $\mathcal{O}$. C_i and D are arbitrary concepts, excluding $\top$, $\bot$ and literals of the form X or $\neg X$ where X is not in the signature of $\mathcal{O}$, $\mathbf{C}$ is a possibly empty disjunction of arbitrary concepts. $C \equiv D$ is syntactic sugar for $C \sqsubseteq D$ and $D \sqsubseteq C$, as is $=nR.D$ for $\geq nR.D \sqcap \leq nR.D$. Domain and range axioms are GCIs so that $Domain(R, C)$ means $\exists R.\top \sqsubseteq C$, and $Range(R, C)$ means $\top \sqsubseteq \forall R.C$. The negation normal form of D is $\mathsf{nnf}(D)$.

The transformation ensures that concept names that are in the signature of $\mathcal{O}$ only appear in axioms of the form $X \sqsubseteq A$ or $X \sqsubseteq \neg A$, where X is some concept name *not* occurring in the signature of $\mathcal{O}$. Note that the structural transformation does not use structure sharing[3].

Axiom Length The definition of laconic justifications uses the notion of the length of an axiom. Length is defined as follows: For X, Y a pair of concepts or roles, A a concept name, and R a role, the length of an axiom is defined as follows:

$$|X \sqsubseteq Y| := |X| + |Y|$$
$$|X \equiv Y| := 2(|X| + |Y|),$$
$$|Sym(R)| = |Trans(R)| := 1$$

where

$$|\top| = |\bot| := 0,$$
$$|A| = |\{i\}| = |R| := 1,$$
$$|\neg C| := |C|$$
$$|C \sqcap D| = |C \sqcup D| := |C| + |D|$$
$$|\exists R.C| = |\forall R.C| := 1 + |C|$$
$$| \geq nR.C| = | \leq nR.C| := 1 + |C|$$

[3]For example, given $\top \sqsubseteq C \sqcup \exists R.C$, two new names should be introduced, one for each use of C, to give $\{\top \sqsubseteq X_0 \sqcup X_1, X_1 \sqsubseteq \exists R.X_2, X_2 \sqsubseteq C\}$. The preclusion of structure sharing ensures that the different positions of C are captured.

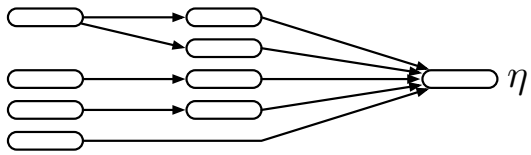

Figure 1: A schematic of a one-to-many proof for a laconic justification

Note: This definition is slightly different from the usual definition, but it allows cardinality axioms such as $A \sqsubseteq \leq 2R.C$ to be weakened to $A \sqsubseteq \leq 3R.C$ without increasing the length of the axiom.

Definition 2 (Laconic Justification) *Let $\mathcal{O}$ be an ontology such that $\mathcal{O} \models \eta$. $\mathcal{J}$ is a* laconic justification *for η over $\mathcal{O}$:*

1. *$\mathcal{J}$ is a justification for η in $\mathcal{O}^\star$*
2. *$\delta(\mathcal{J})$ is a justification for η in $\delta(\mathcal{O}^\star)$*
3. *For each $\alpha \in \delta(\mathcal{J})$ there is no α' such that*
   (a) $\alpha \models \alpha'$ and $\alpha' \not\models \alpha$
   (b) $|\alpha'| \leq |\alpha|$
   (c) $\delta(\mathcal{J}) \setminus \{\alpha\} \cup \delta(\{\alpha'\})$ is a justification for η in $\delta(\mathcal{O}^\star)$

Example

Proofs

The standard proof theoretical notion of proof (or *derivation* as we shall call it) is a list of sentences such that the inclusion of each sentence is in accordance with some *inference rule* from a designated set of rules (the proof theory). Typically, some of the sentences will be assumptions, *premises*, that is, sentences that were substantively derived,[4]. If we want to be somewhat strict, we could restrict premises to those elements of the derivation which are not derivable from the empty set, and, if removed from the derivation, one (or at least one) will be the *conclusion* of the derivation.

From the point of view of end user explanation,

Understanding Justifications

Proofs for Laconic Justifications

Definition 3 (Laconic Justification Proof) *A laconic justification proof for a justification $\mathcal{J}$ for an entailment η in $\mathcal{O}$ is a weakly connected directed acyclic graph $G = (V, E)$ such that $\mathcal{J} \subseteq V \subset \mathcal{J}^\star$ and either, $G = (\{\eta\}, \{\langle \eta, \eta \rangle\})$ or,*

1. *$v = \eta$ is the one and only sink node in G*
2. *$V' = \mathcal{J}$ is the exact set of source nodes in G*
3. *For a given node, the set of predecessor nodes are a justification for the node over $\mathcal{J}^\star$*

A schematic of a Laconic Justification Proof is shown in Figure 1

[4]Obviously, the phrasing "substantively dervied" is a bit circular, but harmless at this stage

Conclusion

References

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

Horridge, M.; Parsia, B.; and Sattler, U. 2008. Laconic and precise justifications in OWL. In *ISWC 08 The International Semantic Web Conference 2008, Karlsruhe, Germany*.

Kalyanpur, A. 2006. *Debugging and Repair of OWL Ontologies*. Ph.D. Dissertation, The Graduate School of the University of Maryland.

Motik, B.; Shearer, R.; and Horrocks, I. 2007. Optimized reasoning in description logics using hypertableaux. In *Proc. of the 21st Int. Conf. on Automated Deduction (CADE-21)*, volume 4603 of *Lecture Notes in Artificial Intelligence*, 67–83. Springer.

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 International Joint Conference on Artificial Intelligence*.