Lemmas for Justifications in OWL*

Matthew Horridge, Bijan Parsia, Ulrike Sattler

The University of Manchester
Oxford Road, Manchester, M13 9PL
{matthew.horridge|bparsia|sattler}@cs.man.ac.uk

1 Introduction

Over the past few years there has been a significant amount of interest in the area of debugging and repairing of OWL ontologies. The process of debugging an ontology is necessary in the same way that debugging programme code is necessary – that is, debugging takes place in order to eradicate faults. In terms of ontology debugging, the faults manifest themselves as *undesirable entailments.* In particular, the entailment that a concept is unsatisfiable is almost always undesired. However, undesirable entailments are not restricted to unsatisfiable concepts. Other entailments, such as certain subsumptions between concepts, might be unintended and contrary to the modeler's understanding of the domain, and thus, undesirable. Ontology debugging is the process of finding the causes of an undesirable entailment, understanding these causes, and modifying the ontology so that the undesirable entailment no longer holds.

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. Justifications are minimal subsets of an ontology that are sufficient for an entailment to hold. Virtually all mainstream ontology editors such as Protégé-4, Swoop, and Top Braid Composer provide support for generating justifications as explanations for arbitrary entailments. Justifications have proved enormously useful for understanding and debugging ontologies. In [1], Kalyanpur presents a user study which showed that the availability of justifications had a significant positive impact on the ability of users to successfully diagnose and repair an ontology. Recently, justifications have been used for debugging very large ontologies such as SNOMED [2], where the size of the ontology prohibits efficient manual debugging.

In the same way that debugging software requires an understanding of *why* errors in the code occur, it is necessary to understand *why* undesirable or unexpected entailments arise in a buggy ontology. Without this understanding, it

* The user studies that form part of this work were approved by the University of Manchester Senate Ethics Committee. Reference 0723

is essentially impossible to devise any kind of reasonable repair strategy for an ontology. However, people can find it difficult to understand certain justifications. Indeed, as will be seen later, there is evidence that some justifications are very difficult, or even impossible, for a wide range of people to understand. The ability to understand a justification is affected by many factors ranging from presentation techniques through to the interplay between the various types of axioms in the justification. This paper focuses on the latter aspect. The work presented in this paper focuses on taking justifications that are difficult to understand, and choosing subsets of these justifications that can be replaced with simpler summarising entailments, such that the result is an easier to understand justification. *Note that it is highly unlikely that this service should be directly exposed to end users. Instead, it is envisioned that a debugging tool will make multiple calls to this service in order to build proof structures to present to users.*

2 Preliminaries

Throughout this paper, the following nomenclature is used:

α	an axiom
$\mathcal{O}$	an ontology
η	an entailment
$\mathcal{J}$	a justification
$\mathcal{S}$	a sets of axioms
$\mathcal{S}^\star$	the deductive closure of $\mathcal{S}$
$Sig(X)$	the signature of X

A and B are used as atomic concept names, C, D, E are used as (possibly complex) concepts, R and S as role names. This paper focuses on OWL and OWL 2 and their rough syntactic variants $\mathcal{SHOIN}(\mathcal{D})$ [3] and $\mathcal{SROIQ}$ [4] respectively. For the purposes of this paper, an *ontology* is regarded as a finite set of $\mathcal{SROIQ}$ axioms $\{\alpha_0, \ldots, \alpha_n\}$. An axiom is of the form of $C \sqsubseteq D$ or $C \equiv D$, where C and D are (possibly complex) concept descriptions, or $S \sqsubseteq R$ or $S \equiv R$ where S and R are (possibly inverse) roles. It should be noted that OWL contains a significant amount of syntactic sugar, such as $DisjointClasses(C, D)$, $FunctionalObjectProperty(R)$ or $Domain(R, C)$. **Signature** The signature of a concept expression, axiom, or set of axioms is the set of concept, role, individual and datatype names appearing in the concept expression, axiom, or set of axioms. $Sig(X)$ denotes the signature of X. **Consistency** A set of axioms $\mathcal{S}$ is *consistent* if there exists an interpretation that satisfies every axiom in $\mathcal{S}$ (i.e. there exists a model of $\mathcal{S}$). **Justifications** A justification [1, 5, 6] for an entailment in an ontology is a *minimal* set of axioms from the ontology that is sufficient for the entailment to hold. The set is minimal in that the entailment does not follow from any proper subset of the justification. More precisely,

Definition 1 (Justification). *For an ontology $\mathcal{O}$ and an entailment η where $\mathcal{O} \models \eta$, 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 $\mathcal{J}' \subsetneq \mathcal{J}$, then $\mathcal{J}' \not\models \eta$. 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 $\mathcal{J}' \not\models \eta$.*

Consider the following ontology, $\mathcal{O} = \{1\colon A \sqsubseteq B,\ \ 2\colon A \sqsubseteq \exists R.A,\ \ 3\colon D \equiv \exists R.B,\ \ 4\colon A \sqsubseteq F,\ \ 5\colon B \sqsubseteq D\}$ which entails $A \sqsubseteq D$. There are two justifications for $A \sqsubseteq D$, the first being $\{1, 2, 3\}$ and the second being $\{1, 5\}$. Notice that if any one of the axioms is removed from any of these justifications, then the remaining set of axioms no longer supports the entailment.

3 Understanding Justifications

Through personal experience and through the observation of users working with justifications, the intuition that some justifications can be very difficult or even impossible for people to understand has emerged. In order to verify this intuition, we conducted a user study. The aim of the study was to gather sufficient data on the difficulty of understanding justifications in order to develop, calibrate, and validate a model for predicting how easy or difficult it would be for people to understand a given justification.

3.1 User Study

Participants Generally speaking, people who view justifications of entailments do so within the context of an ontology development environment. Users request justifications for entailments during the editing or browsing process when they encounter undesirable entailments or entailments that they do not understand. The target population of the study was therefore people who have had experience of either browsing, editing, or using OWL ontologies. The target population did not include OWL neophytes, or people with a very limited understanding of OWL who would find it difficult to read and interpret OWL axioms. Given the target population, it was reasonable for the study sample to comprise staff and students from computer science departments, Semantic Web researchers, and people from other research communities who are familiar with OWL.

Procedure We collected a corpus of justifications for entailments found in published OWL ontologies. The entailments were either unsatisfiable concept entailments ($A \equiv \bot$) or subsumption entailments between named concepts ($A \sqsubseteq B$). We selected a subset of the corpus which seemed difficult for people to understand (e.g., we removed trivial justifications such as those consisting of the entailment itself). We then *expanded* the set of justifications using various substitution lemmas that we speculated would make justifications easier to understand. In total this provided a pool of 100 justifications which were used in the study. In order to remove biasing effects from domain knowledge (or lack of domain knowledge), justifications were obfuscated by replacing the names of entities with alpha-numeric identifiers.

A random selection of justifications was presented to each participant. For each justification, we recorded the time taken for the participant to claim that they had understood (or had not understood) the justification. The “think aloud protocol” was used in order for the study facilitator to determine whether or not the participant had, in fact, understood the justification. We also recorded the participant's ranking on how easy or difficult the justification was to understand using a six point *Likert* scale: {'Very easy'=1, 'Easy'=2, 'Neither easy or difficult'=3, 'Difficult'=4, 'Very difficult'=5, 'Impossible'=6}.

Results A total of 12 people participated in the study. The participants' experience with OWL ranged from less than 6 months to over 4 years. Some participants only had experience in browsing ontologies, while other participants develop OWL tools. The total number of rankings, a ranking being an instance of a viewing of a justification, was 227 (an average of 18.9 rankings per participant). Figure 1 shows a plot of ranking versus time (in seconds). Each point represents a ranking. A rank of 1 corresponds to "very easy to understand", and a rank of 6 corresponds to "impossible to understand". Excluding ranking 6 ("impossible to understand") the general trend indicates that when participants took longer to understand a justification they perceived it to be more difficult to understand. It is noticeable that, in many cases the time spent trying to understand justifications that were deemed impossible to understand (raking 6), is less than the time spent trying to understand very difficult justifications. This indicates that participants gave up trying to understand a justification very soon, perhaps because it simply seemed to complicated, or, after some effort they thought that they would never understand the justification. In fact, it was common for participants who could not understand a particular justification to ask the question, "Is this explanation correct?", thereby implying that they doubted the ability of the system to generate sound justifications. Out of the 227 rankings, 69 (30%) corresponded to being "difficult to understand" through to "impossible to understand" (35 rankings, (15%) corresponded to being "impossible" to understand). Interestingly, all of the "impossible to understand" rankings were rankings of unmodified, *naturally occurring*, justifications. In summary, there were a significant number of "difficult" to "impossible" to understand justifications, which indicates that understanding justifications is a real problem.

3.2 Factors that Affect Understanding

Data from the user study provided insight into how people understand justifications and why they find certain justifications difficult to understand. Broadly speaking, the following reasons were identified as being important.

When people work through justifications they typically perform obvious syntactic transformation of axioms and spot simple patterns to reach intermediate conclusions. For example, a user might work through a justification, such as $\{1, 2, 3\}$, from the Example in Section 2, as follows: Axioms 1 and 2 entail $A \sqsubseteq \exists R.B$, and this result, in conjunction with Axiom 3 entails $A \sqsubseteq D$ (the entailment). Note that the user must spot a suitable intermediate inference step, understand how it arises, and understand the part it plays in the whole justification. Justifications that contain a lot of information, and a lot of intermediate inference steps, are difficult to understand. In particular, the number of different *types* of axioms and concept expressions that the justification contains plays an

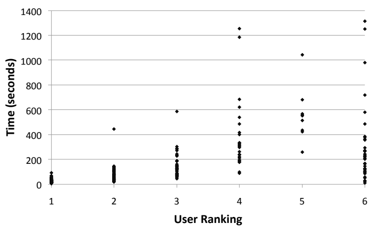

Fig. 1: User Ranking vs. Time

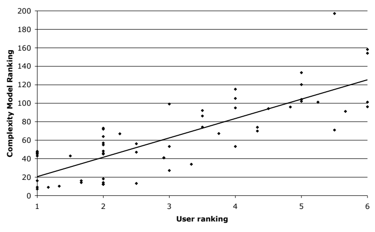

Fig. 2: User Ranking vs. Model

important part. Justifications whose intermediate inference steps arise due to the interaction between many different types of axioms or concept expressions are hard to understand.

Justifications that contain unfamiliar patterns of axioms are difficult to understand. Consider the following ontology,$\mathcal{O} = \{1\colon A \equiv \forall R.C, 2\colon Domain(R, A), 3\colon E \sqsubseteq F\}$, which is derived from a real ontology[1] and was presented to some of the study participants as part of a justification. This ontology entails $E \sqsubseteq A$. The reason for this is that Axioms 1 and 2 entail $\top \sqsubseteq A$. During the study, it was observed that many of the participants (including participants with many years of experience with OWL, and even reasoner developers) did not realise, or neglected to see, that $A \equiv \forall R.C$, coupled with $Domain(R, A)$, entails $\top \sqsubseteq A$. Many of the participants had not encountered this "pattern of axioms" before. They therefore had difficulty in realising what these axioms entail, and their significance in the context of the complete justification. There are, of course, other such patterns of axioms that occur in justifications that people find difficult to spot or understand.

A commonality between the two points detailed above is that subsets of a justification can result in entailments that can be viewed as "steps" or "intermediate entailments". When trying to understand justifications, it is necessary for people to spot and understand these intermediate entailments. In terms of how complex a justification is for people to understand, the number of significant intermediate entailments, and for any given intermediate entailment, the number of different types of axioms and concept expressions that give rise to the entailment, has an effect. What counts as a significant intermediate entailment is an open question; however, this basic idea of steps gives rise to the notion *lemmas for justifications*.

4 A Model for Predicting Complexity

Based on the data obtained from the study, we developed a simple model in order to predict how difficult it is for people to understand a given justification. The model is also used as a tool for identifying the significant intermediate inference steps in a justification. The model is composed of two parts: 1) A

[1] This example was taken from an ontology about movies, which was originally posted to the Protege-OWL mailing list.

"structural" complexity measure, which estimates the complexity based on the number of different types of axioms and concept expressions in a justification; 2) A "phenomena" based complexity measure, which increases the complexity of a justification when certain patterns of axioms or "phenomena" occur in the justification.

For the sake of brevity only an informal description and summary of the model is given here:

An "axiom and concept expression type" component estimates complexity based on the syntax of justifications. It predicts complexity based on the number of different types of axioms and concept expressions in a justification. Different types of axioms, and concept expressions have different weightings. These weightings are based on the data obtained in the user study. For example, inverse role axioms have a heavy weighting in the complexity model because many participants found justifications that contained inverse properties difficult to understand. Likewise, subclass axioms that have a complex concept expression on the left hand side are weighted heavier than subclass axioms that have a concept name on the left hand side.

A "signature flow" component, reflects the degree of spreading of terms in the signature of the justification across axioms in the justification (how much the signature "flows" across the axioms in the justification). This component was introduced as an indicator for justifications where there are many intermediate steps that reuse the same axioms to justify their conclusions. In some sense, it reflects the "tangling" of the justification.

A "universal implication" component increases the computed complexity if there are any subclass axioms that have a left hand side of $\forall R.C$, or equivalent concept axioms which state that $\forall R.C$ is equivalent to some other concept expression. This component was motivated by the fact that many participants failed to realise that $\forall R.C$ subsumes the class of individuals that have no R successor.

A "general concept inclusion" component increases the complexity of a justification as the number of General Concept Inclusions present in laconic versions of the justification increases. GCIs in laconic justifications are usually a direct indicator of an intermediate inference step that needs to be spotted.

A "synonym of top" component increases the complexity if the justification has any entailed synonyms of $\top$ that are not explicitly asserted. It was found that study participants were generally surprised by this kind of entailment, with most of them having trouble spotting it.

It should be noted that various weightings are used throughout the model. The purpose of these weightings is to allow individual components of the model to be altered for tuning purposes. In a user setting, it might also be possible to tune out certain components, such as the "universal implication" component, when users start to feel comfortable spotting certain patterns of axioms. In a similar vein, the model could be augmented with additional components should other patterns be discovered in the future. In any case, the model presented here is a first approximation, and, as will be seen, is satisfactory for the purpose of lemmatising justifications.

A version of the model was implemented using weightings based on data from the study. Figure 2 shows a plot of the user rankings that were assigned to justifications against complexity as predicted by the model. The model prediction has a linear relationship with the rankings provided by study participants. In fact, there is a correlation of 0.8 between the two variables, which indicates a strong correlation. In what follows the model is used as a first approximation for predicting how difficult it is for people to understand a justification, which in turn, is used to defined lemmas as the notion of intermediate inference steps.

5 Lemmas for Justifications

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 definition is given, then a more precise definition is given in Definition 3.

Informally, a set of lemmas Λ 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 $\mathcal{J}' = (\mathcal{J} \setminus \mathcal{S}) \cup \Lambda$ 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 Λ that can lemmatise a justification $\mathcal{J}$. These restrictions prevent counter-intuitive lemmatisations, an example of which will be given below. Before these restrictions are discussed, it is necessary to introduce the notion of a *tidy* set of axioms.

Definition 2 (Tidy sets of axioms). *A set of axiom $\mathcal{S}$ is* tidy *if $\mathcal{S} \not\models \top \sqsubseteq \bot$, $\mathcal{S} \not\models A \sqsubseteq \bot$ for all $A \in Sig(\mathcal{S})$, and $\mathcal{S} \not\models \top \sqsubseteq A$ for all $A \in Sig(\mathcal{S})$.*

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 $\mathcal{S} \models \top \sqsubseteq A$).

The restrictions mandate that a set of lemmas Λ 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 axioms in Λ, 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 Λ). 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 intuitively lemmas for $\mathcal{J} \models A \sqsubseteq \bot$. Similarly, counter-intuitive results arise if lemmas are drawn from *inconsistent* sets of axioms, or sets of axioms that contain synonyms for $\top$. The definition of lemmas below, Definition 3, therefore only allows Λ to contain axioms that are drawn from (i) the deductive closures of *tidy* subsets of $\mathcal{S} \subseteq \mathcal{J}$ (ii) the set of direct/explicit synonyms of $\top$ or $\bot$ with respect to $\mathcal{S}$.

In what follows, δ is the 'well known' structural transformation originally defined in [8] (with a version of the rewrite rules for description logic syntax given in [7]). 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. $\mathcal{T}^\star$ 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 a person to understand—the larger the value the more complex. In this case, the previously described model is used to provide a complexity rating.

Definition 3 (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 Θ be the set of tidy (Definition 2) subsets of $(\mathcal{S} \cup \delta(\mathcal{S}))$. Let Ω be the set of consistent subsets of $(\mathcal{S} \cup \delta(\mathcal{S}))$. Let*

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

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

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

6 Computing Lemmas

Algorithm 1 is a practical algorithm for computing lemmatised justifications. The algorithm requires two sub-routines: The ComputeJustifications sub-routine returns the justifications for an entailment that holds in a set of axioms—any "off the shelf" implementation of a justification finding service may be used here. The ComputeCandidateLemmas sub-routine computes a set of axioms that are entailed by *tidy subsets* (Definition 2) of the set of input axioms. In the implementation described in this paper, the ComputeCandidateLemmas subroutine computes lemmas that in themselves have a low complexity and hence users find easy to understand. In particular, the implementation computes lemmas of the form $A \sqsubseteq B$, $C \sqsubseteq A$, $A \sqsubseteq \exists R.B$, $Domain(R, A)$, $A(a)$, $(\exists R.A)(a)$ and $S \sqsubseteq R$. The implementation of the routine generates these axioms and checks that they are entailed by *tidy* subsets of the set of input axioms. It should be noted that Algorithm 1 is non-deterministic—the output depends on the complexity model that is used to determine whether one justification is simpler than another, and also on the GetCandidateLemmas subroutine. The algorithm always terminates.

Algorithm 1 LemmatiseJustification

```
Function-1: LemmatiseJustification($J, \eta$)
1: $S \leftarrow J \cup$ ComputeCandidateLemmas($J, \eta$) $\setminus \{\eta\}$
2: $justs \leftarrow$ ComputeJustifications($S, \eta$)
3: $c_1 \leftarrow$ ComputeComplexity($J, \eta$)
4: $L \leftarrow J$
5: for $J' \in justs$ do
6:     $c_2 \leftarrow$ ComputeComplexity($J', \eta$)
7:     if $c_2 < c_1$ then
8:         $L \leftarrow J'$
9: return $L$
```

6.1 Implementation Evaluation

In order to demonstrate that it is practical to compute lemmatised justifications, the 1 algorithm was implemented in Java using the latest version of the OWL API in conjunction with the Pellet reasoner. Justifications (over 450 for entailments of the form $A \sqsubseteq B$, $A \sqsubseteq \bot$ and $A(a)$) whose predicted complexity was greater than 100.0 (roughly corresponding to 'Difficult', 'Very difficult' or 'Impossible'), were then selected for lemmatisation from the ontologies, shown in Table 3. For each justification a lemmatised version of the justification that had a predicted complexity of less than 50.0 (roughly corresponding to a 'very easy' or 'easy' ranking) was computed. The implementation was run on a laptop with a 2.16GHz Intel Core Duo Processor, with the Java Virtual Machine allocated 1GB RAM.

Figure 4 shows a plot of the initial justification complexity against the time required for computing lemmatised justifications. It is clear to see that, in most cases, lemmatised justifications can be computed in under two seconds. The maximum time required was 4.5 seconds. It is noticeable that the plot exhibits a clustering effect. This is due to the fact that in certain ontologies there were many justifications with a similar complexity score. For example the clustering between 100 and 150 is caused by justifications extracted from the Chemical ontology.

ID	Ontology	Expressivity	Logical Axioms	Justifications
1	Generations	$\mathcal{ALCOIF}$	38	59
2	Nautilus	$\mathcal{ALCHF}(\mathcal{D})$	38	10
3	People	$\mathcal{ALCHOIN}$	108	150
4	Periodic-table	$\mathcal{ALU}$	214	378
5	University	$\mathcal{SOIN}(\mathcal{D})$	52	12
6	Economy	$\mathcal{ALCH}(\mathcal{D})$	1625	1383
7	Transportation	$\mathcal{ALCH}(\mathcal{D})$	1157	188
8	MiniTAMBIS	$\mathcal{SHOIN}$	173	65
9	Earthrealm	$\mathcal{SHOIN}(\mathcal{D})$	2546	262
10	Chemical	$\mathcal{ALCHF}(\mathcal{D})$	114	424

Fig. 3: Ontologies used for Testing

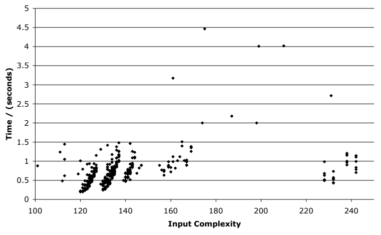

Fig. 4: Lemmatisation Time

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

Fig. 5: A justification for $\mathsf{Person} \sqsubseteq \bot$

Entailment : $\mathsf{Person} \sqsubseteq \bot$

- $\boldsymbol{\mathsf{Person} \sqsubseteq \neg\mathsf{Movie}}$
- $\top \sqsubseteq \mathsf{Movie}$
    - $\forall \mathsf{hasViolenceLevel}.\bot \sqsubseteq \mathsf{Movie}$
        - $\forall \mathsf{hasViolenceLevel}.\bot \sqsubseteq \mathsf{RRated}$
            - $\boldsymbol{\mathsf{RRated} \equiv (\exists \mathsf{hasScript}.\mathsf{ThrillerScript}) \sqcup (\forall \mathsf{hasViolenceLevel}.\mathsf{High})}$
        - $\mathsf{RRated} \sqsubseteq \mathsf{Movie}$
            - $\boldsymbol{\mathsf{RRated} \sqsubseteq \mathsf{CatMovie}}$
            - $\boldsymbol{\mathsf{CatMovie} \sqsubseteq \mathsf{Movie}}$
    - $\exists \mathsf{hasViolenceLevel}.\top \sqsubseteq \mathsf{Movie}$
        - $\boldsymbol{\mathsf{Domain}(\mathsf{hasViolenceLevel}, \mathsf{Movie})}$

Fig. 6: A schematic of a series of lemmatised justifications arranged into a tree. The root justification is for $\mathsf{Person} \sqsubseteq \bot$

7 An Example

Figure 5 shows a justification $\mathcal{J}$ for $\mathsf{Person} \sqsubseteq \bot$. A lemmatisation of this justification, along with lemmatisations of the justifications for the lemmas over $\mathcal{J}$ is shown in Figure 6. Axioms in $\mathcal{J}$ are shown in bold, while lemmas are shown in a plain typeface. In this example, $\mathcal{J}$ was initially lemmatised to give $\mathcal{J}' = \{\top \sqsubseteq \mathsf{Movie}, \mathsf{Person} \sqsubseteq \neg\mathsf{Movie}\}$. Since $\top \sqsubseteq \mathsf{Movie}$ is lemma in $\mathcal{J}'$, another justification for this lemma was computed over $\mathcal{J}$ and then itself lemmatised. This process was repeated to automatically build up the tree shown in Figure 6. It should be noted that this presentation is for illustrative purposes and to give a flavour of the kinds of lemmas introduced into a justification, it is not necessarily intended for end users.

8 Related Work

In [9], a sequent calculus is used as the basis for explaining subsumption in $\mathcal{ALC}$. The proofs produced by this approach explicitly reference the inference rules that are used to go from one step to the next, and in this regard are fairly close to formal proofs and not in the spirit of justifications. Borgida briefly mentions the idea of sub-steps and weakenings as ways of deriving higher quality explanations. In [10], Schlobach uses interpolation to explain subsumption in $\mathcal{ALC}$. He searches for interpolants that have particular syntactic and semantic properties. Schlobach calls these interpolants *illustrations*, and uses them to help explain how one subsumption follows from another. The basic motivations are to make explanations easier to understand. Lingenfelder [11] and Huang [12] tackle the problem of presenting machine generated proofs to humans. In both cases, they attempt to address the problem that machine generated proofs are difficult for humans to understand. Lingenfelder remarks that even natural deduction proofs are at too low a level for human understanding, and that their length causes difficulty in seeing "the important steps" and therefore hinders understanding. Huang also argues that natural deduction proofs are also at too low a level, and develops *ND style proofs* that are at a higher level of abstraction. Interestingly, Lingenfelder sketches the idea of grouping proof steps together and applying lemmas. He also points out that it is necessary to distinguish between trivial steps and more complicated steps, possibly with use of a model. There has been a significant amount of work on predicting the complexity of understanding and the ease of maintainability of software. In particular, seminal work by McCabe [13] was followed by a plethora of work. Some of the inspiration and ideas for the properties of the complexity model presented here were drawn from this work.

9 Conclusions and Future Work

A wide range of people can find justifications for entailments in OWL difficult to understand. The work that has been presented in this paper attempts to begin to address this problem through the use of lemmas for justifications.

A model that predicts how difficult it is for people to understand a justification has been defined. The model was calibrated and validated against data from a user study. This model has been used as input into a definition of lemmas for justifications. The definition specifies that a lemmatisation of a justification results in another justification that is easier to understand according to the notion of justification complexity. Initial empirical results indicate that it is feasible to compute lemmatised justifications for entailments from published ontologies.

The next major challenge is to design and evaluate services that make use of lemmatised justifications for building proof structures that are ultimately aimed at end users. More studies will be needed to evaluate these mechanisms and to show that they can be beneficially integrated into ontology development environments and user workflows.

References

1. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. PhD thesis, The Graduate School of the University of Maryland (2006)
2. Baader, F., Suntisrivaraporn, B.: Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In: KR-MED. (2008)
3. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. J. of Web Semantics (2003)
4. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: KR 2006. (2006)
5. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. J. Autom. Reasoning (1995)
6. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: IJCAI. (2003)
7. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in owl. In: ISWC. (2008)
8. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. J. of Symbolic Computation (1986)
9. Borgida, A., Franconi, E., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F.: Explaining $\mathcal{ALC}$ subsumption. In: ECAI. (2000)
10. Schlobach, S.: Explaining subsumption by optimal interpolation. In: JELIA. (2004)
11. Lingenfelder, C.: Structuring computer generated proofs. In: IJCAI. (1989)
12. Huang, X.: Reconstructing proofs at the assertion level. In: CADE 12. (1994)
13. McCabe, T.J.: A complexity measure. In: IEEE Trans. On Software Eng. (1976)