The Justificatory Structure of OWL Ontologies

Samantha Bail, Bijan Parsia, Ulrike Sattler

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

Abstract. Current ontology development tools offer debugging support by presenting justifications for entailments of OWL ontologies. In many cases even a single entailment may have many distinct justifications, and justifications for distinct entailments may be critically related. We call the set of relations between multiple justifications the *justificatory structure* of an ontology. A restricted analysis of justificatory structure has already been successfully exploited to reduce effort when debugging ontologies with large numbers of unsatisfiable classes by identifying *root* unsatisfiable classes. In this paper we present a preliminary analytical framework for the justificatory structure of an ontology and explore possible applications.

1 Introduction

Explanation support significantly improves the user experience when working with OWL ontologies. With regard to the task of debugging it is often impossible to find the cause of an erroneous entailment, such as an unsatisfiable class, without any tools that guide the user to the source of the error [9]. Different methods have been developed to assist ontology developers in understanding and repairing the errors, such as pinpointing, model exploration, and justifications [12, 1].

Justifications are minimal subsets of the ontology that are sufficient for an entailment to hold. Explanation support through justifications is currently provided by ontology development tools such as Protégé 4. Current research mainly focuses on making individual justifications easier to understand, for example through defining *fine-grained* justifications [10, 5] and analysing patterns [2]. Another important aspect that is being explored is the efficient computation of justifications, in particular finding all justifications for an entailment [11, 13], and dealing with inconsistent ontologies [6]. However, there has been relatively little research into using justifications as a way to obtain further information about an ontology.

We are now interested not only in what justifications can tell us about an entailment, but also how the relationships between justifications affect the entire ontology, the understanding of the user, and potential repair strategies in the debugging process. Measuring metrics in order to assess properties like the cohesion of an ontology has been the focus of previous research [3, 16], and different

approaches have been implemented in user-oriented tools [14, 15]. However, the existing frameworks only consider the class hierarchy and axiom metrics, and do not make use of the information provided by the justifications for entailments of the ontology.

In this paper we introduce the *justificatory structure* of OWL ontologies, that is, metrics describing the occurrences of multiple justifications, as well as dependencies and other relations between justifications in an ontology. We show that these structural properties describe important and useful information to the users, which can help them understand entailments and their origins by providing deeper insight into the ontology. In order to show the usefulness of this approach, we outline potential application areas and explain which aspects of the justificatory structure can be used in the respective context.

2 Preliminaries

2.1 OWL Syntax and Notation

In this paper, we use the OWL Manchester Syntax[1] and following notation: $\mathcal{O}$ for an ontology, $A, B, \ldots$ for class names, r and s for property names, and a for an individual. Axioms can be of the form *(Class: C SubClassOf: D)* or *(Class: C EquivalentTo: D)*, where C and D are class expressions that are built from class and property names.[2] An entailment of an ontology $\mathcal{O}$ is written as $\mathcal{O} \models \eta$.

2.2 Justifications

Typically, not all axioms in an ontology are needed to cause an entailment to hold. In many cases a small subset of the ontology is already sufficient. Working with these subsets when trying to understand the reason for an entailment has shown to be much easier than having to deal with the full ontology [9]. Justifications are minimal subsets of an ontology $\mathcal{O}$ that cause an entailment η to hold. They are are defined as follows [5]:

Definition (Justifications) $\mathcal{J}$ is a justification for $\mathcal{O} \models \eta$ if $J \subseteq \mathcal{O}, J \models \eta$ and, for all $J' \subset J$, it holds that $J' \not\models \eta$.

The *unsatisfiability* of a class is a particularly relevant entailment in the debugging process, as it commonly represents a modelling error in the ontology. A class A is unsatisfiable wrt. an ontology $\mathcal{O}$ if $\mathcal{O} \models$ *(A SubClassOf: owl:Nothing)*. An ontology that contains unsatisfiable classes is called *incoherent*, as shown in

[1] http://www.w3.org/TR/owl2-manchester-syntax

[2] For legibility and space reasons, we omit several parts of the Manchester syntax, including declarations and the leading *Class:*. The complete examples can be accessed at http://owl.cs.manchester.ac.uk/explanation/owled2010.

this simple example where class C is unsatisfiable:

$$\begin{aligned}\mathcal{O} = \{&C\ SubClassOf:\ A\ and\ D\\ &A\ SubClassOf:\ E\ and\ B\\ &B\ SubClassOf:\ not\ D\ and\ r\ some\ D\\ &F\ SubClassOf:\ r\ only\ A\\ &D\ SubClassOf:\ s\ some\ owl{:}Thing\} \models C\ SubClassOf:\ owl{:}Nothing\end{aligned}$$

Here, there exists only one justification for the unsatisfiability of C, which is the set of the first three axioms. It is obvious that the error is much easier to spot once we know which axioms to focus on, rather than examining the whole ontology. Often, there exist multiple justifications for one entailment, and it is not unusual to have ten or more justifications per entailment, as shown in table 2.

With respect to debugging, it is often the task to find a *repair* that "breaks" the entailment. This can be achieved by removing one axiom from each justification from the ontology [8]. Since axioms are removed from the ontology, a repair can affect not only the entailment in question, but also other entailments. In order to have minimal unwanted impact on the ontology, it is helpful to find a repair that is as small as possible. This means that axioms occurring in multiple justifications are suitable candidates for removal.

While incoherence can be regarded as a modelling error that needs to be repaired, a large number of actively used ontologies do in fact contain unsatisfiable classes. This does not cause any further problems, unless statements are added that lead to a *contradiction* in the knowledge base. For example, if the axiom *(Individual: a Types: C)* was added to the example above, the ontology would be rendered *inconsistent*, since it would require an instance of an uninstantiable class.

The term fine-grained describes justifications whose axioms do not contain any superfluous information [5]. *Laconic* justifications are particularly helpful with respect to understanding justifications, as they allow the user to focus on the relevant parts of an axiom. There can be more or less laconic than regular justifications.

3 The Justificatory Structure of an Ontology

We identify a set of relations that represent different aspects of the justificatory structure of an OWL ontology. These aspects can be classified into: quantifiable properties of justifications in an ontology (metrics), information about the syntactic relations of justifications, and semantic relationships between justifications.

3.1 Multiple Justifications

Multiple justifications can be regarded in the context of single entailments, as well as for multiple distinct entailments. In the following we mainly discuss

the occurrence of multiple justifications for a single entailment, namely inferred atomic subsumptions and unsatisfiable named classes.

There are two interesting aspects when dealing with multiple justifications, which we denote as *coping* and *exploiting*. Firstly, when seen from a debugging point of view, we can ask: how can we cope with this potentially large number of justifications? Is it possible to find useful information about their interactions, which could simplify the repair process? Based on findings from preliminary experiments with 16 ontologies (all available from the TONES[3] repository), it is easy to see that multiple justifications do occur in ontologies.

While the set of ontologies presented in table 1 is not representative of all ontologies, it illustrates different properties and phenomena that can occur in ontologies of various sizes and expressivities. Table 2 shows the average number of regular and laconic justifications (AvgR, AvgL), as well as the respective maxima (MaxR, MaxL) measured in our experiments. It also lists the number of unsatisfiable classes (UC) and how many of these are root unsatisfiable (RUC). The number of regular justifications per entailment in our test set differs

#	Ontology	Expressivity	Axioms	Entailments
1	MGEDOntology	$\mathcal{ALEOF}(\mathcal{D})$	4679	2
2	DOLCE Lite	$\mathcal{SHIF}$	536	3
3	Mini Tambis	$\mathcal{ALCN}$	400	65
4	Nautilus	$\mathcal{ALCHF}$	172	10
5	Generations	$\mathcal{ALCOIF}$	60	24
6	Mereology	$\mathcal{SHIN}$	80	2
7	Relative Places	$\mathcal{SHIF}$	130	7
8	Cell	$\mathcal{EL}++$	14743	11
9	People + Pets	$\mathcal{ALCHOIN}$	370	33
10	University	$\mathcal{SOIN}(\mathcal{D})$	92	10
11	Numerics	$\mathcal{SHIF}(\mathcal{D})$	478	3098
12	Earth Realm	$\mathcal{ALCHO}$	1613	2751
13	Economy	$\mathcal{ALCH}(\mathcal{D})$	2330	51
14	Programmes	$\mathcal{SHIF}(\mathcal{D})$	560	51
15	Adolena	$\mathcal{SRIQ}$	415	3
16	Chemical	$\mathcal{ALCHF}$	192	43

Table 1. The 16 ontologies used in our experiments

strongly, ranging from exactly one justification (e.g. Mini Tambis, Nautilus) to 24 (DOLCE Lite). The average number of regular justifications per entailment is 1.8, with an average size of 3.3 axioms. Note that in some cases, entailments with single regular justifications have multiple laconic ones. The extreme cases in particular, where up to 68 laconic justifications are obtained for a single entailment, show that it is necessary to develop methods that help users cope with such a large number of justifications.

[3] http://owl.cs.manchester.ac.uk/repository

#	Ontology	AvgR	MaxR	AvgL	MaxL	UC	RUC
1	MGEDOntology	1.00	1	1.00	1	0	0
2	DOLCE Lite	1.00	1	24.00	68	0	0
3	Mini Tambis	1.00	1	1.88	4	30	5
4	Nautilus	1.00	1	1.00	1	0	0
5	Generations	1.00	1	0.92	1	0	0
6	Mereology	1.00	1	1.00	1	0	0
7	Relative Places	1.00	1	1.00	1	0	0
8	Cell	1.09	2	1.09	2	0	0
9	People + Pets	1.09	2	1.36	4	1	1
10	University	1.20	3	2.10	6	9	6
11	Numerics	1.27	5	1.27	5	2	2
12	Earth Realm	1.29	4	1.29	4	2	2
13	Economy	1.29	2	1.29	2	51	34
14	Programmes	1.98	9	1.29	3	2	2
15	Adolena	2.00	3	2.00	3	0	0
16	Chemical	9.86	26	9.86	26	37	2

Table 2. Metrics of justifications for the chosen ontologies

Exploiting refers to a different aspect of multiple justifications: is it possible to utilize this phenomenon in order to obtain information about the ontology itself? In which way do the relationships between justifications affect other properties of the ontology, and vice versa? For example, regarding the results from our experiments, we would like to learn why there are such significant differences in the numbers of justifications for each ontology. From these considerations also follows the question of how this information can then be made accessible to the user, suited to the required task.

3.2 Metrics

Simple statistics about the justifications found in an ontology can provide an insight into its structure and connectedness, which will be discussed in section 4. These statistics include the number and size of regular justifications for a single entailment, the number and size of laconic justifications for a single entailment and their respective ratios.

3.3 Syntactic Relations Between Justifications

Subset Relationships One of the most important syntactic relationships is the containment of one justification in another. This property has been utilised in the definition of root and derived unsatisfiable classes, which are relevant for the debugging and repair process.

Presenting justifications to the user and distinguishing root and derived unsatisfiable classes has shown to drastically reduce user effort when debugging an ontology that has multiple unsatisfiable classes [9, 8]. The intuitive definition

is as follows: Derived unsatisfiable classes depend on the unsatisfiability of another class (the *parent* of the derived unsatisfiable class) and may be fixed (i.e. made satisfiable) by simply repairing this parent. It is possible for a derived unsatisfiable class to depend on multiple parent classes. Root unsatisfiable classes are classes whose unsatisfiability does not depend on another class. The precise definition translates this into a statement about subsets of justifications [4].

Definition (Root and derived unsatisfiable classes) A class C that is unsatisfiable with respect to an ontology $\mathcal{O}$ is *derived unsatisfiable* if there exists a justication J for $\mathcal{O} \models$ *(C SubClassOf: owl:Nothing)*, and a justication J' for $\mathcal{O} \models$ *(D SubClassOf: owl:Nothing)* such that $J' \subset J$. An unsatisfiable class that is not a derived unsatisfiable class is known as a *root unsatisfiable* class.

In the following example, $\mathcal{O}$ entails the unsatisfiability of both C and A, $J_1 =$*{C SubClassOf: D, C SubClassOf: not D}* and $J_2 = \mathcal{O}$ being the respective justifications:

$$\begin{aligned}\mathcal{O} = \{&A\ SubClassOf:\ r\ some\ C\\ &C\ SubClassOf:\ D\\ &C\ SubClassOf:\ not\ D\} \models A\ SubClassOf:\ owl:Nothing\end{aligned}$$

A is a derived unsatisfiable class, as its justification J_2 is a strict superset of the justification J_1 for the unsatisfiability of C. We can also say that J_1 *causes* the unsatisfiability, while J_2 *propagates* it. By repairing C's unsatisfiability (for example by removing the third axiom from the set), class A will also be repaired. In terms of debugging unsatisfiable classes, this shows that by repairing the root unsatisfiable classes first all the derived unsatisfiable classes may be fixed at the same time.

In some ontologies, such as Tambis,[4] which contains 144 unsatisfiable classes, it has been shown that nearly all of the derived unsatisfiabilities (111 in Tambis) could be repaired by simple fixing a small number of root unsatisfiable classes (only 3 in the case of Tambis) [9]. This dependency can be clearly used in helpful tool support and depends largely on the structure of the ontology.

Equality This concerns the case where multiple justifications for different entailments contain exactly the same axioms. This simply means that the same set of axioms has multiple entailments and happens to be a justification, i.e. minimal, for all these entailments. $\mathcal{J}_1 =$ *{A SubClassOf: B, B SubClassOf: C and D}* for example is a justification for two entailments that are atomic subsumptions, namely *(A SubClassOf: C)* and *(A SubClassOf: D)*. When looking at laconic justifications only, we obtain two distinct justifications for the entailments and the equality does no longer hold. This provides information about the modelling as well as potential redundancies in the ontology, and it clearly shows the relevance of laconic justifications for the comprehensibility of explanations.

[4] http://www.cs.man.ac.uk/∼stevensr/tambis

Intersection As a more general case of subset relations, intersection provides a starting point when developing a repair strategy for breaking an entailment. Again, we only consider syntactical overlap here, i.e. multiple justifications sharing a common axiom. Removing only one axiom that occurs in the overlapping parts of multiple justifications for a single entailment can lead to a repair that has less impact on the rest of the ontology. This is desirable, as repairs should be minimal and ideally only affect the entailment in question [8, 12].

3.4 Semantic Relationships

Entailment It is possible for justifications to entail each other. This can be both unidirectional ($J_1 \models J_2$, $J_2 \not\models J_1$) and bidirectional ($J_1 \models J_2$, $J_2 \models J_1$). A special case of entailment is a subset relationship, as a set of axioms naturally entails its subsets.

Masking A special case of dependencies between justifications is the phenomenon of masking [7]. This describes the interaction of justifications (and other axioms that are not part of the justification) that conceal the actual number of explanations, as demonstrated in the following example.

$$\begin{aligned}\mathcal{O} = \{&A\ SubClassOf\colon\ B\ and\ not\ B\ and\ C\\ &C\ SubClassOf\colon\ D\ and\ not\ D\}\\ \models\ &A\ SubClassOf\colon\ owl\colon Nothing\end{aligned}$$

The only justification is $J_1 = \{A\ SubClassOf\colon B\ and\ not\ B\ and\ C\}$, and there are no root / derived relationships. If we attempt to break the entailment, for example by removing *(not B)* from the justification, it still holds because of the unsatisfiability now being derived from the second axiom. This gives us another justification for the entailment, namely $J_2 = \mathcal{O}$, which is clearly a superset of J_1. This case is not captured by the above definition for derived justifications, as a superset of a justification for the same entailment is by definition not a justification (due to the minimality constraint). However, we lose valuable information if this dependency of J_2 on J_1 is not pointed out to the user in the repair process.

Shared Cores Masking Shared cores describe a particular type of masking, where the justifications have parts that are structurally equal. This is illustrated by the following example:

$$\begin{aligned}\mathcal{O} = \{&A\ SubClassOf\colon\ B\ and\ not\ B\ and\ C\\ &A\ SubClassOf\colon\ B\ and\ not\ B\}\\ \models\ &A\ SubClassOf\colon\ owl\colon Nothing\end{aligned}$$

The part *(and C)* can be removed from the justification $J_1 = \{A\ SubClassOf\colon B\ and\ not\ B\ and\ C\}$, as it is not relevant for the entailment to hold. This leads to the laconic version of J_1, which is syntactically equal to $J_2 = \{A\ SubClassOf\colon$

B and not B}. If this phenomenon is pointed out to the user, they can easily see that there exists only a single reason for an entailment rather than several, which they can then focus on in the repair process.

3.5 Classifying Justificatory Structure

Defining or classifying the justificatory structure of an ontology with respect to some metric allows to investigate how the structure affects the ontology and vice versa. We can potentially categorise different types of justificatory structure intuitively, based on their complexity: a weak justificatory structure exhibits only a small average number of mostly disjoint justifications, such as one justification per entailment. An ontology with a strong (complex) justificatory structure comprises a large number of justifications for each entailment and a high degree of interconnectivity (intersections, subset relationships, entailment) between them. Extensive experiments will allow us to identify the aspects of justificatory structure and their respective weights that are most suitable for specifying a metric to classify it.

4 Application

In this section, we provide an overview of potential applications of analysing the justificatory structure. These cover a wide range of potential users from ontology engineers to reasoner developers, as well as both task-specific and global usage in the ontology development process.

4.1 Debugging

One of the main tasks that explanation deals with is debugging support in the ontology engineering process. First of all, by making use of the information about dependencies between justifications, the user can be guided to understand the cause of an entailment. The information can then be used to provide a suitable repair strategy, that allows the user to amend the entailment without causing unwanted changes to the ontology. We use the abstract term information here, as there exist different levels of interaction with the user: we can provide raw data, such as $J_1 \subset J_2$, which can already be helpful for experienced users. By embedding this information into a more user-friendly representation and exploiting it in tools, such as a visualisation interface, understanding dependencies and their impact on entailments can be made more accessible to the user.

4.2 Ontology Comprehension

Explanation support for ontology comprehension can be considered different from using justifications for debugging, as it is less task-based and success is harder to define: what exactly does understanding the ontology mean? One way of defining ontology comprehension is based on the ability to answer questions

relating to information in the ontology, as previously shown in a user study [1]. We believe that structural information about the ontology in a suitable representation can help the user understand the dependencies between axioms, the modelling choices that were made, and even help them to spot non-logical errors that cannot be detected with the help of a reasoner.

4.3 Analytics

In addition to debugging and ontology comprehension, the suggested metrics can also provide useful data when analysing ontologies and developing tools. In addition to metrics such as the expressivity of an ontology and the number of classes and axioms, the justificatory structure offers a way of describing and classifying ontologies. This relates to the notion of *axiomatic richness*, which describes the expressiveness and use of *interesting*, non-trivial class expressions in an ontology.

Thus far, axiomatic richness has no formal definition and is more of an abstract concept than a measurable property. As an example, taxonomic ontologies containing only trivial axioms of the form *(A SubClassOf: B)* are commonly regarded as axiomatically *weak*. A simple indicator for axiomatic richness could be a large average number of justifications for entailments. Reasoner development and testing can be regarded as another potential application area of the justificatory structure. We can ask: does a certain type of justificatory structure make reasoning harder? Again, this hypothesis has to be tested in more extensive experiments.

5 Conclusion and Future Work

In this paper, we have presented a framework for analysing the various dependencies between justifications in OWL ontologies, which are believed to offer useful structural information about an ontology. We have shown that there exists a number of interactions between justifications, such as syntactic overlap and entailment. The different aspects of this justificatory structure of an ontology were grouped into syntactical connections, semantic relations and metrics. Services using the justificatory structure in the ontology development process could support users with debugging tasks, assist in understanding ontologies and provide metrics for classifying ontologies.

For future work, we aim to define the different aspects of the justificatory structure of an ontology more clearly. In the long term, algorithms and services will be developed that generate and use the data, which can then be presented to the user in a way tailored to the respective task. Examining different forms of visualisation for this purpose offers another extension to the topic discussed in this paper.

References

1. J. Bauer, U. Sattler, and B. Parsia. Explaining by example: Model exploration for ontology comprehension. In *Description Logics*, 2009.
2. O. Corcho, C. Roussey, L. M. V. Blázquez, and I. Pérez. Pattern-based OWL ontology debugging guidelines. In *WOP*, 2009.
3. A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. A theoretical framework for ontology evaluation and validation. In *SWAP*, 2005.
4. M. Horridge, J. Bauer, B. Parsia, and U. Sattler. Understanding entailments in OWL. In *OWLED*, 2008.
5. M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in OWL. In *International Semantic Web Conference*, pages 323–338, 2008.
6. M. Horridge, B. Parsia, and U. Sattler. Explaining inconsistencies in OWL ontologies. In *SUM*, pages 124–137, 2009.
7. M. Horridge, B. Parsia, and U. Sattler. Justification masking in OWL. In *Description Logics*, 2010.
8. A. Kalyanpur, B. Parsia, E. Sirin, and B. Grau. Repairing unsatisfiable concepts in OWL ontologies. In *ESWC*, pages 170–184. Springer, 2006.
9. A. Kalyanpur, B. Parsia, E. Sirin, and J. Hendler. Debugging unsatisfiable classes in OWL ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(4):268–293, 2005.
10. J. Lam, J. Pan, D. Sleeman, and W. Vasconcelos. A fine-grained approach to resolving unsatisfiable ontologies. In *Web Intelligence*. Springer, 2006.
11. S. Schlobach. Diagnosing terminologies. In *AAAI*, 2005.
12. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI*, pages 355–362, 2003.
13. B. Suntisrivaraporn, G. Qi, Q. Ji, and P. Haase. A modularization-based approach to finding all justifications for OWL DL entailments. In *ASWC*, pages 1–15, 2008.
14. S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth, and B. Aleman-Meza. OntoQA: Metric-based ontology quality analysis. In *ICDM*, 2005.
15. A. L. Tello and A. Gómez-Pérez. Ontometric: A method to choose the appropriate ontology. *Journal of Database Management*, 15(2):1–18, 2004.
16. H. Yao, A. Orme, and L. Etzkorn. Cohesion metrics for ontology design and application. *Journal of Computer Science*, 1(1):107–113, 2005.