Laconic and Precise Justifications in OWL

Matthew Horridge, Bijan Parsia, and Ulrike Sattler

School of Computer Science
The University of Manchester
Oxford Road
Manchester
M13 9PL

Abstract. A justification for an entailment in an OWL ontology is a minimal subset of the ontology that is sufficient for that entailment to hold. Since justifications respect the syntactic form of axioms in an ontology, they are usually neither *syntactically* nor *semantically* minimal. This paper presents two new subclasses of justifications—*laconic justifications* and *precise justifications*. Laconic justifications only consist of axioms that do not contain any superfluous "parts". Precise justifications can be derived from laconic justifications and are characterised by the fact that they consist of flat, small axioms, which facilitate the generation of semantically minimal repairs. Formal definitions for both types of justification are presented. In contrast to previous work in this area, these definitions make it clear as to what exactly "parts of axioms" are. In order to demonstrate the practicability of computing laconic, and hence precise justifications, an algorithm is provided and results from an empirical evaluation carried out on several published ontologies are presented. The evaluation showed that laconic/precise justifications can be computed in a reasonable time for entailments in a range of ontologies that vary in size and complexity. It was found that in half of the ontologies sampled there were entailments that had more laconic/precise justifications than regular justifications. More surprisingly it was observed that for some ontologies there were fewer laconic justifications than regular justifications.

1 Introduction

Since the Web Ontology Language, OWL, became a W3C standard, there has been a notable increase in the number of people building, extending and using ontologies. As a result of this, a large number of people have been enticed into using some kind of description logic reasoning service as an aid during the construction and deployment of ontologies. As people have gained in confidence, they have begun to move from creating or using modestly expressive ontologies, through to using richly axiomatised ontologies that exercise the full expressivity of OWL-DL. Experience of delivering a variety of tools to a wide range of users has made it evident that there is a significant demand for editing environments that provide sophisticated editing and browsing support services. In particular, the generation of *justifications* [1] for entailments is now recognised as highly desirable functionality for both ontology development and ontology reuse. A clear

A. Sheth et al. (Eds.): ISWC 2008, LNCS 5318, pp. 323–338, 2008.

demonstration of the need for *practical* explanation services that provide justifications was demonstrated by the fact that many users switched from Protégé 3.2 to Swoop purely for the benefits of automatic explanation facilities.

The ability to obtain justifications for entailments was originally exposed to the masses in the ontology editing and browsing tool Swoop [2]. Since then, other OWL tools, such as Protégé-4 [3], OWLSight,[1] and Top Braid Composer[2] have adopted the ability to generate these justifications, showing the importance of explanations to end users.

Intuitively, a justification is a set of axioms from an ontology that is sufficient for an entailment to hold. A key aspect of justifications is that they operate on the level of *asserted* axioms. That is, the axioms in a justification directly correspond to axioms that have been asserted in the ontology in which the entailment holds. Therefore, if an ontology contains "long" axioms, for example, ones containing many complex class expressions, then there may be *parts* of axioms in a justification that are not required for the entailment that is supported by the justification. For example, consider the set of axioms $\mathcal{J} = \{A \sqsubseteq B \sqcap C, C \sqsubseteq D, A \sqsubseteq \neg D\}$ which entails $A \sqsubseteq \bot$. The right hand side of first axiom in the set contains the conjunct B. However, this conjunct does not play any part in causing A to be unsatisfiable. In this sense, if this set of axioms is a justification for $A \sqsubseteq \bot$ then it could be more "fine-grained"—it should be somehow possible to indicate that only "part" of the first axiom is required for the entailment $A \sqsubseteq \bot$ to hold. Justifications that only contain parts of axioms that are relevant for the entailment to hold, have been referred to as "fine-grained" justifications [4], and also "precise justifications" [5].

While there is a general feeling that fine-grained justifications should only consist of the *parts of axioms* that are relevant to the entailment in question, there have not been any attempts to produce a rigourous formalisation of these kinds of justifications. This means that while it is cut and dried as to what exactly a justification is, the same cannot be said for fine-grained justifications. In particular, it is not clear what it means to talk about *parts of axioms*. Not only does this make it difficult for implementers to be sure they can generate fine-grained justifications in a sound and complete manner, it also makes it difficult to compare the approach taken by one system in generating fine-grained justifications to the approach taken by other systems.

The purpose of this paper is to identify the desirable characteristics of fine-grained justifications, propose a formal definition for these types of justifications, and show how they can be computed for OWL-DL. In order to demonstrate that computing these justifications according to this definition is feasible, an algorithm is provided, which is evaluated on a sample of published ontologies.

2 Preliminaries

Throughout this paper, the following nomenclature is used.

[1] http://pellet.owldl.com/ontology-browser/
[2] http://www.topbraid.org

α	an axiom; subscripts and primes are used to denote different axioms
$\|\alpha\|$	refers to the length of α
$\mathcal{O}$	an ontology
$\delta(\mathcal{O})$	a set of axioms that is the result of a structural transformation on $\mathcal{O}$
η	an arbitrary entailment that is assumed to hold in some ontology or set of axioms
$\mathcal{O}^\star$	the deductive closure of an ontology $\mathcal{O}$
$(\delta(\mathcal{O}))^\star$	the deductive closure of the structural transformation of $\mathcal{O}$
$\mathcal{O}^+$	a subset of the deductive closure of an ontology $\mathcal{O}$
$\mathcal{S}$	a set of sets of axioms
$\mathcal{J}$	a justification
$\mathcal{L}$	a description logic, e.g. $\mathcal{ALC}$, $\mathcal{SHOIQ}$

A, B, C, D, E are used as concept names, R and S as role names, n and n' are place holders for positive integers. $\mathcal{T}$ refers to a T-Box, $\mathcal{R}$ a Role-Box and $\mathcal{A}$ an A-Box.

Given an ontology, $\mathcal{O}$, and a description logic $\mathcal{L}$ the *deductive closure* of $\mathcal{O}$, is written as $\mathcal{O}^\star_\mathcal{L}$, where $\mathcal{O}^\star_\mathcal{L} = \{\alpha \in \mathcal{L} \mid \mathcal{O} \models \alpha\}$. In other words the deductive closure contains *all* well formed $\mathcal{L}$-axioms that are entailed by the ontology $\mathcal{O}$. When it is clear from the context, the subscript $\mathcal{L}$ is dropped.

An axiom α' is said to be *weaker* than another axiom, α iff $\alpha \models \alpha'$ and $\alpha' \not\models \alpha$.

OWL and Description Logics. This paper focuses on OWL-DL or its rough syntactic variant $\mathcal{SHOIN}(\mathcal{D})$ [6], but the approach can be applied to other description logics such as $\mathcal{SROIQ}$, which will underpin the next version of OWL.

For the purposes of this paper, an *ontology* is regarded as a finite set of $\mathcal{SHOIN}$ 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. OWL contains a significant amount of syntactic sugar, such as $DisjointClasses(C, D)$, $FunctionalObjectProperty(R)$ or $Domain(R, C)$, however, these kinds of axioms can be represented using subclass axioms [6].

Justifications. A justification [1,7,8] 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. It should be noted that there may be several, potentially overlapping, justifications for a given entailment.

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 in $\mathcal{O}$ if $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \eta$ and if $\mathcal{J}' \subsetneq \mathcal{J}$ then $\mathcal{J}' \not\models \eta$.*

3 Motivation for Fine-Grained Justifications

Justifications have proved to be very useful in general. However, there are at least four reasons that motivate the investigation of fine-grained justifications:

1. An axiom in a justification can contain irrelevant parts. Consider $\mathcal{O} = \{B \sqsubseteq C \sqcap D, D \sqsubseteq E\} \models B \sqsubseteq E$. Clearly, $\mathcal{O}$ is a justification for $B \sqsubseteq E$, but the first conjunct in the first axiom is irrelevant for this entailment and might *distract* a user from identifying the relevant parts. It is arguable that focusing a user's attention on the relevant parts of an axiom can make it easier for them to understand a justification.
2. A justification can conceal relevant information. Consider $\mathcal{O} = \{B \sqsubseteq \neg C \sqcap D, B \sqsubseteq C \sqcap \neg D\} \models B \sqsubseteq \bot$. B is unsatisfiable for two distinct reasons, but there is only a single justification for it (containing both axioms from $\mathcal{O}$). If this entailment is broken by deleting one of the axioms then modelling errors could be introduced—it may be that the repaired ontology should contain $B \sqsubseteq D \sqcap \neg C$. We refer to this condition as *internal masking*.
3. Justifications can mask relevant axioms. Consider $\mathcal{O} = \{B \sqsubseteq D \sqcap \neg D \sqcap C, B \sqsubseteq \neg C\} \models B \sqsubseteq \bot$. There is no justification in $\mathcal{O}$ for $B \sqsubseteq \bot$ that includes $\{B \sqsubseteq \neg C\}$, yet, it clearly plays a role in entailing the unsatisfiability. Users working with justifications would most likely be unaware of this. We refer to this condition as *external masking*
4. Multiple justifications can conceal a fine-grained core. In certain cases there may be multiple justifications for an entailment but fewer fine-grained justifications for the same entailment. Consider $\{A \sqsubseteq B \sqcap C, B \sqsubseteq D\}$ and $\{A \sqsubseteq B \sqcap F, B \sqsubseteq D\}$ as two justifications for $A \sqsubseteq B$. There is just one fine-grained justification: $\{A \sqsubseteq B, B \sqsubseteq D\}$. Besides making the entailment easier to understand, this scenario might also indicate modelling errors or redundancies.

A common point running through all of the above is the issue of repair. Since a justification is a subset of an ontology, and consists of *asserted* axioms, it is relatively straightforward, and intuitive, to devise a repair for the ontology that breaks the entailment in question: Given some undesired entailment η that holds in an ontology $\mathcal{O}$, and a set of justifications $\mathfrak{J}$ for η, a simple method of breaking the entailment is to choose one axiom from each justification $\mathcal{J} \in \mathfrak{J}$ and remove these chosen axioms from $\mathcal{O}$. However, from the above examples, it should be fairly clear that when working with "regular" justifications there is a potential to "over repair" an ontology so that more entailments are lost than is necessary. In this sense, it is desirable that any definition of fine-grained justifications should result in justifications that make it easy to devise and enact a minimal and consistent repair of an ontology. Ideally, the underlying repair process should mimic the intuitive process of repair when working with regular justifications—one axiom from each fine-grained justification should be identified as a candidate for removal from the ontology or some suitable variant.

4 Related Work

In [5], Kalyanpur et al. propose an algorithm for computing "precise" justifications. The algorithm rewrites axioms into smaller axioms in order to obtain the relevant "parts".

The ontology editor Swoop [2] features the ability to "strike out" irrelevant parts of axioms in a justification. However, this is based on a heuristic approach, and while it is very efficient, and is strongly expected to be sound, it is incomplete.

In [4], Lam presents "fine grained" justifications for $\mathcal{ALC}$ with general TBoxes. A tableaux reasoning technique, which is an extension of the technique from Meyer et al. [9] and Baader and Hollunder [10] is used.

In [8], Schlobach and Cornet focus on computing explanations for unsatisfiable classes. They define the concepts of *MUPS* (Minimal Unsatisfiability Preserving Sub-TBoxes) and MIPS (Minimal Incoherence Preserving Sub-TBoxes), which are special cases of justifications. Schlobach and Cornet describe a procedure that *syntactically generalises* all of the axioms in each MIPS to produce a generalised TBox that contains smaller axioms which are responsible for any unsatisfiable classes.

Deng et al. [11] take a novel approach to "measuring inconsistencies in description logic ontologies" by using results obtained from Game Theory. Although no details are provided, Deng claims that the technique could easily be extended to pinpoint the "proportions" of axioms that are responsible for an unsatisfiable class, via the use of clause pinpointing.

Finally, in [12] Baader et al. pinpoint axioms for entailments in the description logic $\mathcal{EL}$. Although the work is not concerned with fine-grained justifications, the $\mathcal{EL}$ subsumption algorithm uses a normalisation procedure that flattens axioms and makes them smaller. These smaller axioms could be used to indicate the parts of axioms responsible for an entailment.

A common aim of all previous approaches for computing fine-grained justifications is to determine the *parts* of axioms that are responsible for a particular entailment. However, none of these approaches define exactly what they mean by *parts* of axioms. Moreover, each approach is specific to a particular implementation technique and is defined in an *operational* sense. This means that it is generally unclear as to what exactly constitutes a fine-grained justification. As a consequence, it is unclear as to whether any one approach for computing fine-grained justifications would result in the obtaining the same set of fine-grained justifications for an entailment when compared to another approach.

In summary, a *general definition of fine-grained justifications* is needed. Ideally, such a definition would not be tied to a particular DL. This definition would then permit the evaluation and comparison of algorithms for computing fine-grained or "precise" justifications and, it would make it possible to investigate the underlying problem in a thorough way.

5 Laconic and Precise Justifications Defined

There appear to several desirable properties that a definition for fine-grained justifications should satisfy. In particular:

- **Minimality** Each axiom in a fine-grained justification should, in some sense, be as *small* as possible—each axiom should only capture the parts of the asserted form that are required for the entailment in question to hold.

– **Repair** As with regular justifications, fine-grained justifications should suggest as simple a repair as possible. Ideally, removing an axiom from a fine-grained justification should generate a repair that is minimal in terms of lost entailments.

In what follows a definition of fine-grained justifications is proposed. This definition consists of two parts: 1) a definition of what we term *laconic justifications*, which informally are justifications that do not contain any superfluous parts; 2) a definition of what we term *precise* justifications, that can be derived from laconic justifications, and are such that their axioms are flat, small and semantically minimal. Precise justifications are primarily geared towards repair.

5.1 δ–The Structural Transformation

The definition of laconic justifications below uses $\delta(\mathcal{J})$, where δ is a satisfiability preserving *structural transformation* on $\mathcal{J}$ that removes all nested descriptions and hence produces axioms that are as small and flat as possible. An appropriate transformation, a version of which is shown below, is the well known structural transformation described in Plaisted and Greenbaum [13] and used in [14].

$$
\begin{aligned}
\delta(\mathcal{O}) &:= \textstyle\bigcup_{\alpha \in \mathcal{R} \cup \mathcal{A}} \delta(\alpha) \cup \bigcup_{C_1 \sqsubseteq C_2 \in \mathcal{T}} \delta(\top \sqsubseteq \mathsf{nnf}(\neg C_1 \sqcup C_2)) \\
\delta(D(a)) &:= \delta(\top \sqsubseteq \neg\{a\} \sqcup \mathsf{nnf}(D)) \\
\delta(\top \sqsubseteq \mathbf{C} \sqcup D) &:= \delta(\top \sqsubseteq A'_D \sqcup \mathbf{C}) \cup \textstyle\bigcup_{i=1}^{n} \delta(A'_D \sqsubseteq D_i) \text{ for } D = \prod_{i=1}^{n} D_i \\
\delta(\top \sqsubseteq \mathbf{C} \sqcup \exists R.D) &:= \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq \exists R.A'_D\} \cup \delta(A'_D \sqsubseteq D) \\
\delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.D) &:= \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq \forall R.A'_D\} \cup \delta(A'_D \sqsubseteq D) \\
\delta(\top \sqsubseteq \mathbf{C} \sqcup \geq nR.D) &:= \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq \geq nR.A'_D\} \cup \delta(A'_D \sqsubseteq D) \\
\delta(\top \sqsubseteq \mathbf{C} \sqcup \leq R.D) &:= \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq \leq nR.A'_D\} \cup \delta(A'_D \sqsubseteq D) \\
\delta(A'_D \sqsubseteq D) &:= \delta(A'_D \sqsubseteq D) \text{ (If } D \text{ is of the form } A \text{ or } \neg A) \\
\delta(A'_D \sqsubseteq D) &:= \delta(\top \sqsubseteq \neg A'_D \sqcup D) \text{ (If } D \text{ is } \textit{not} \text{ of the form } A \text{ or } \neg A) \\
\delta(\beta) &:= \beta \text{ for any other axiom}
\end{aligned}
$$

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].

[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.

5.2 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:

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

where

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

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.

5.3 Laconic Justifications

With a suitable structural transformation, δ, and the notion of axiom length in hand, laconic justifications can be defined. (Recall that $\mathcal{O}^\star$ is the deductive closure of $\mathcal{O}$)

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 $(\delta(\mathcal{O}))^\star$*

Intuitively, a laconic justification is a justification where all axioms only contain sub-concepts (parts) that are relevant for the entailment in question, and moreover, these sub-concepts (parts) are *as weak as possible.*

5.4 Notes on Definition 2

$\mathcal{O}^\star$—the deductive closure of $\mathcal{O}$. It is apparent from Definition 2(1) that laconic justifications for an entailment in an ontology may be drawn from the deductive closure of that ontology. Therefore, unlike regular justifications, laconic justifications are not specific to the asserted axioms in an ontology. This ensures that it is possible to capture the internal and external masking cases highlighted in Section 3.

***δ*—a structural transformation.** The primary use of a structural transformation in Definition 2 is to transform a justification into an equi-satisfiable set of axioms, $\delta(\mathcal{J})$, where each axiom does not have any nested complex descriptions (each axiom is "flattened out") and each axiom is as small as possible. These axioms can be thought of as a representation of all of the "parts" of the axioms in $\mathcal{J}$. Thus, ensuring that the axioms in $\delta(\mathcal{J})$ are as weak as possible ensures that all "parts" of axioms in a laconic justification are as weak as possible.

Applying the transformation to a justification results in two possibilities:

1. $\delta(\mathcal{J})$ is *not* a justification for η—it is a superset of a justification for η because $\mathcal{J}$ consists of axioms that contains parts or strong parts that are not required for $\mathcal{J} \models \eta$. Hence condition 2 of Definition 2 is violated and $\mathcal{J}$ is not laconic.
2. $\delta(\mathcal{J})$ *is* a justification for η, which implies that all sub-concepts of all axioms (in their existing or in a weakened form) are required for $\mathcal{J} \models \eta$. Hence, if each axiom in $\delta(\mathcal{J})$ is as weak as possible, as dictated by condition 3 of Definition 2, then $\mathcal{J}$ is laconic.

Example 1. Consider the following ontology $\mathcal{O} = \{\alpha_1 \colon A \sqsubseteq B, \alpha_2 \colon B \sqsubseteq D, \alpha_3 \colon A \sqsubseteq B \sqcap C\} \models A \sqsubseteq D$. There are two justifications for $\mathcal{O} \models A \sqsubseteq D$, $J_1 = \{\alpha_1, \alpha_2\}$ and $J_2 = \{\alpha_2, \alpha_3\}$. By Definition 2, J_1 *is* a laconic justification since $\delta(J_1) = \{\top \sqsubseteq X_0 \sqcup X_1, X_0 \sqsubseteq \neg A, X_1 \sqsubseteq B, \top \sqsubseteq X_2 \sqcup X_3, X_2 \sqsubseteq \neg B, X_3 \sqsubseteq D\}$ neither of these axioms can be weakened further without lengthening them or without resulting in $\mathcal{J}_1 \not\models A \sqsubseteq D$. Conversely, J_2 *is not* a laconic justification since after performing the structural transformation to obtain $\delta(\mathcal{J})$ a superset of a justification for η is obtained.

Definition 3. ***(Precise Justification)***

Let $\mathcal{O}$ be an ontology such that $\mathcal{O} \models \eta$. Let $\mathcal{J}$ be a justification for $\mathcal{O} \models \eta$ and let $\mathcal{J}' = \delta(\mathcal{J})$. $\mathcal{J}'$ is precise *with respect to $\mathcal{J}$ if $\mathcal{J}$ is a laconic justification for $\mathcal{O} \models \eta$.*

Intuitively, a precise justification is a version of a laconic justification where the axioms contained in the precise justification are as flat, small and as weak as possible. In essence, a precise justification is a "repair friendly" version of a laconic justification. Note that *a precise justification is precise with respect to a laconic justification*—a justification cannot in itself be precise.

Lemma 1. *All laconic justifications can be converted to precise justifications by means of the structural transformation, δ.*

5.5 Repair

Although "repair" is not the primary subject of this paper, it should be noted that the motivation behind Definition 3 is based on the notion that it should be possible to generate a repair that makes semantically minimal changes to an ontology. Because, by definition, a precise justification contains axioms that are maximally flat, small and weak, it is only necessary to remove one of these

axioms in order to generate a minimal repair. A semantically minimal repair will be generated if an axiom of the form $X \sqsubseteq A$ or $X \sqsubseteq \neg A$, where X is any named introduced in the structural transformation and A is a concept name occurring in the signature of $\mathcal{O}$, is removed (hence the requirement that δ produces axioms where concept names from the signature of $\mathcal{O}$ only occur in the aforementioned pattern).

In order to generate a semantically minimal repair for an entailment, laconic justifications for the entailment should be first generated, then precise justifications for these laconic justifications should be computed. The precise justifications can then be used to guide the process of axiom selection to indicate the parts of asserted axioms that should be removed.

6 Filtering Laconic Justifications

Since laconic justifications are defined with respect to the deductive closure of an ontology, it is not difficult to see that there could be many laconic justifications for an entailment. For example, given $\mathcal{O} = \{C \sqsubseteq D \sqcap \neg D \sqcap E, A \sqsubseteq B\}$, laconic justifications for $\mathcal{O} \models C \sqsubseteq \bot$ include $\{C \sqsubset D \sqcap \neg D\}$ and $\{C \sqsubseteq D, C \sqsubseteq \neg D\}$. It is noticeable that both of these justifications are somewhat structurally or syntactically related to the asserted axioms in $\mathcal{O}$. However, Definition 2, also admits other laconic justifications such as $\{C \sqsubseteq B \sqcap \neg B\}$ (since $C \sqsubseteq B \sqcap \neg B$ is also in the deductive closure of $\mathcal{O}$). Despite the fact that this is a valid laconic justification, it is arguable that justifications of this ilk, which could be considered to be syntactically irrelevant or "incidental", are not of general interest to an ontology modeller who is trying to understand the reasons for an entailment.

In order to focus on syntactically relevant laconic justifications, a filter on the deductive closure, called $\mathcal{O}^+$, is introduced. $\mathcal{O}^+$ is essentially a *representative* of the deductive closure of an ontology, which gives rise to *preferred* laconic justifications.

Definition 4. *($\mathcal{O}^+$ completeness) Let $\mathcal{O}$ be an ontology, $\mathcal{O}^+$ a set of axioms such that $\mathcal{O} \subseteq \mathcal{O}^+ \subseteq \mathcal{O}^\star$, and η such that $\mathcal{O} \models \eta$. $\mathcal{O}^+$ is* complete for *η and $\mathcal{O}$ if for $\mathfrak{J}$ the set of all laconic justifications for η w.r.t. $\mathcal{O}^+$, for any $\mathcal{O}'$ such that $\mathcal{O}' \subseteq \mathcal{O}^+$, if $\mathcal{O}' \not\models \mathcal{J}$ for all $\mathcal{J} \in \mathfrak{J}$, then $\mathcal{O}' \not\models \eta$.*

Completeness ensures that the laconic justifications can be used for a simple repair: if an ontology is weakened so that it entails none of the laconic justifications, then it no longer entails η.

The exact details of how to construct a suitable $\mathcal{O}^+$ depend somewhat on how laconic justifications will be used. For example, an application that presents laconic justifications to end users may well prefer justifications that maintain conjunctions or disjunctions. For example given $\mathcal{O} = \{C \sqsubseteq D \sqcap \neg D \sqcap E\}$, it might be preferable to generate $\{C \sqsubseteq D \sqcap \neg D\}$ as opposed to $\{C \sqsubseteq D, C \sqsubseteq \neg D\}$ as a laconic justification for $C \sqsubseteq \bot$. The reverse may be true if generating laconic

justifications to display in a repair tool, where smaller axioms might suggest a more appropriate repair.

In what follows an $\mathcal{O}^+$ is specified for the description logic $\mathcal{SHOIQ}$ ($\mathcal{O}^+_{\mathcal{SHOIQ}}$) so as to capture the sort of laconic (fine-grained) OWL-DL justifications in play in the literature. Importantly, $\mathcal{O}^+_{\mathcal{SHOIQ}}$ produces laconic justifications that are syntactically relevant—there is a direct correspondence between asserted axioms in $\mathcal{O}$ and axioms that appear in laconic justifications, which is essential for usability in tools such as browsers and editors, and directly corresponds with the strikeout feature that is available in Swoop. It should be noted that $\mathcal{O}^+_{\mathcal{SHOIQ}}$ is finite, which means that the set of precise justifications with respect to $\mathcal{O}^+_{\mathcal{SHOIQ}}$ is also finite. Even though it is possibly exponentially larger than $\mathcal{O}$, it will later be seen that it is not necessary to compute $\mathcal{O}^+_{\mathcal{SHOIQ}}$ in its entirety.

Definition 5. *Let* $\mathcal{O}^+_{\mathcal{SHOIQ}} = \{\alpha' \mid \alpha' \in \sigma(\alpha)$ *where* $\alpha \in \mathcal{O}\}$*. We define the mappings* $\sigma(\alpha)$*,* $\tau(C)$ *and* $\beta(C)$ *inductively as follows, where* $X \in \{\tau, \beta\}$ *is used as a meta-variable with* $\overline{\tau} = \beta$*,* $\overline{\beta} = \tau$*,* $\max_\tau = \top$*,* $\max_\beta = \bot$*, and* n' *be the maximum number in number restrictions in* $\mathcal{O}$*:*

$$
\begin{aligned}
\sigma(C_1 \sqcup \cdots \sqcup C_n \sqsubseteq D_1 \sqcap \cdots \sqcap D_m) &:= \{C'_i \sqsubseteq D'_j \mid C'_i \in \beta(C_i), D'_j \in \tau(D_j)\} \\
\sigma(C \equiv D) &:= \sigma(C \sqsubseteq D) \cup \sigma(D \sqsubseteq C) \cup \{C \equiv D\} \\
\sigma(R \sqsubseteq S) &:= \{R \sqsubseteq S\} \\
\sigma(R \equiv S) &:= \{R \sqsubseteq S\} \cup \{S \sqsubseteq R\} \\
\sigma(Trans(R)) &:= \{Trans(R)\} \\
X(A) &:= \{\max_X, A\} \text{ for } A \text{ a concept name or } \{i\} \\
X(C_1 \sqcap \cdots \sqcap C_n) &:= \{C'_1 \sqcap \cdots \sqcap C'_n \mid C'_i \in X(C_i)\} \\
X(C_1 \sqcup \cdots \sqcup C_n) &:= \{C'_1 \sqcup \cdots \sqcup C'_n \mid C'_i \in X(C_i)\}
\end{aligned}
$$

$$
\begin{aligned}
X(\neg C) &:= \{\neg C' \mid C' \in \overline{X}(C)\} \\
X(\exists R.C) &:= \{\exists R.C' \mid C' \in X(C)\} \cup \{\top\} \\
X(\forall R.C) &:= \{\forall R.C' \mid C' \in X(C)\} \cup \{\top\} \\
X(\geq nR.C) &:= \{\geq mR.C' \mid C' \in X(C), n \leq m \leq n'\} \cup \{\max_X\} \\
\tau(\leq nR.C) &:= \{\leq mR.C' \mid C' \in \beta(C), 0 \leq m \leq n\} \cup \{\top\} \\
\beta(\leq nR.C) &:= \{\leq mR.C' \mid C' \in \tau(C), n \leq m \leq n'\} \cup \{\bot\} \\
X(\{j_1 \ldots j_n\}) &:= X(\{j_1\} \sqcup \cdots \sqcup \{j_n\})
\end{aligned}
$$

In essence $\mathcal{O}^+_{\mathcal{SHOIQ}}$ generates weaker, shorter axioms from asserted axioms in $\mathcal{O}$ by, for example, stepwise replacement of sub-concepts with either $\top$ or $\bot$. In fact, $\mathcal{O}^+_{\mathcal{SHOIQ}}$ parallels the well know structural transformation without transforming axioms into negation normal form or clausal form. The benefit of this being a close syntactic correspondence of axioms in $\mathcal{O}^+_{\mathcal{SHOIQ}}$ with axioms in $\mathcal{O}$. From now on the subscript $\mathcal{SHOIQ}$ is dropped so that $\mathcal{O}^+$ refers to $\mathcal{O}^+_{\mathcal{SHOIQ}}$.

7 Computing Laconic Justifications

In order to compute laconic justifications for display in editors such as Swoop and Protégé-4, or for use in other tools such as automatic repair tools, it is necessary

to compute the *preferred laconic justifications*. Recall that these particular kinds of justifications are computed with respect to a representative of the deductive closure of an ontology, namely $\mathcal{O}^+$. Therefore, one of the conceptually simplest methods of computing precise justifications for an entailment η in an ontology $\mathcal{O}$ would be to first compute $\mathcal{O}^+$ directly from $\mathcal{O}$ and then compute justifications with respect to $\mathcal{O}^+$. This would yield a set of justifications that is the *superset* of the set of *preferred* laconic justifications for $\mathcal{O} \models \eta$. The actual set of laconic justifications could then be obtained by some post processing of the justifications that were computed from $\mathcal{O}^+$. However, the size of $\mathcal{O}^+$ is exponential in the size of axioms in $\mathcal{O}$. Since for a given entailment not all axioms and their weakenings will participate in the laconic justifications that are obtained from $\mathcal{O}^+$, computing $\mathcal{O}^+$ for a whole ontology $\mathcal{O}$ could be regarded as being wasteful both in terms of space and time.

Algorithm. *ComputeLaconicJustifications*
Input: $\mathcal{O}$ an ontology, η and entailment such that $\mathcal{O} \models \eta$
Output: $\mathcal{S}$, a set of precise justifications for $\mathcal{O} \models \eta$
1. $\mathcal{S} \leftarrow ComputeOPlusJustifications(\mathcal{O}, \eta)$
2. **for** $S \in \mathcal{S}$
3. **if** $IsLaconic(S, \eta) =$ **false**
4. $\mathcal{S} \leftarrow \mathcal{S} \setminus S$
5. **return** $\mathcal{S}$

Algorithm. *ComputeOPlusJustifications*
Input: $\mathcal{O}$ an ontology, η and entailment such that $\mathcal{O} \models \eta$
Output: $\mathcal{S}$, a set of justifications for $\mathcal{O}^+ \models \eta$
1. $\mathcal{O}' \leftarrow \mathcal{O}$
2. $\mathcal{S} \leftarrow \emptyset$
3. $\mathcal{S}' \leftarrow Justifications(\mathcal{O}', \eta)$
4. **repeat**
5. $\mathcal{S} \leftarrow \mathcal{S}'$
6. **for** $S \in \mathcal{S}$
7. $\mathcal{O}' \leftarrow (\mathcal{O}' \setminus S) \cup ComputeOPlus(S)$
8. $\mathcal{S}' \leftarrow Justifications(\mathcal{O}', \eta)$
9. **until** $\mathcal{S} = \mathcal{S}'$
10. **return** $\mathcal{S}$

Algorithm. *IsLaconic*
Input: J, a justification, η and entailment such that $J \models \eta$
Output: **true** if J is laconic, otherwise **false**
1. $\mathcal{S} \leftarrow Justifications(ComputeOPlus(\delta(J)), \eta)$
2. **return** $\mathcal{S} = \{\delta(\mathcal{J})\}$

It is also tempting to assume that the laconic justifications can efficiently be computed directly from regular justifications without reference to rest of the

ontology. While it may be sufficient to utilise this strategy when implementing a strikeout feature similar to that found in Swoop, this approach would not capture all laconic justifications with respect to $\mathcal{O}^+$. As described in Section 3 point 3, the *external masking* condition means that there could be laconic justifications that would not be found using this technique.

With these points in mind, an optimised algorithm for computing precise justifications, *ComputeLaconicJustifications*, is presented below. Since the algorithm does not require a specific reasoner, or indeed a particular reasoning procedure such as tableau, it is a *Black-Box* algorithm. The algorithm *incrementally* computes the set of all laconic justifications for a given entailment by incrementally computing $\mathcal{O}^+$ from previously found justifications. This yields a set of justifications that is a superset of the laconic justifications for the entailment in question. The algorithm then processes each justification in this set to extract the justifications that are laconic justifications using the $IsLaconic$ subroutine. This subroutine essentially tests whether a justification J is laconic by computing $ComputeOPlus(\delta(\mathcal{J}))$ and then computing laconic justifications from this set of axioms. If the justification is laconic then its singleton set will be equal to the justifications computed from $ComputeOPlus(\delta(\mathcal{J}))$.

The algorithm requires two main subroutines that are not defined below. $Justifications$, which computes the (regular) justifications for an entailment η that holds in some set of axioms (ontology). This subroutine can be implemented using any strategy that computes justifications in accordance with Definition 1. Additionally, the $ComputeOPlus$ subroutine takes a set of axioms and returns a set of axioms that represents $\mathcal{O}^+$ computed from this set of axioms in accordance with Definition 5.

7.1 Performance

In order to evaluate the practicability of computing laconic justifications, the above algorithm and subroutines were implemented using the latest version of the OWL API[4] backed with the Pellet reasoner [15]. This API has clean and efficient support for manipulating an ontology at the level of axioms, and has a relatively efficient and direct wrapper for Pellet. A selection of publicly available ontologies, shown in Table 1 were chosen for number of entailments that hold in them and to provide a range of expressivity.[5]

Each ontology was classified in order to determine the unsatisfiable classes and atomic subsumptions. These kinds of entailments were selected as input to the compute laconic justifications algorithm because they are usually exposed by tools such as Protégé-4 or Swoop and are therefore the kinds of entailments that users typically seek justifications for. For each entailment the time to compute all regular justifications and all laconic justifications was recorded.

[4] `http://owlapi.sourceforge.net`

[5] All of the ontologies used may be found in the TONES ontology repository at `http://owl.cs.manchester.ac.uk/repository`. Entailments include atomic subsumptions and unsatisfiable classes.

Table 1. Ontologies used in experiment

ID	Ontology	Expressivity	Axioms	No. Entailments
1	Generations	$\mathcal{ALCOIF}$	38	24
2	Economy	$\mathcal{ALCH}(\mathcal{D})$	1625	51
3	People+Pets	$\mathcal{ALCHOIN}$	108	33
4	MiniTambis	$\mathcal{ALCN}$	173	66
5	Nautilus	$\mathcal{ALCHF}$	38	10
6	Transport	$\mathcal{ALCH}$	1157	62
7	University	$\mathcal{SOIN}$	52	10
8	PeriodicTableComplex	$\mathcal{ALU}$	58	366
9	EarthRealm	$\mathcal{ALCHO}$	931	543
10	Chemical	$\mathcal{ALCHF}$	114	44
11	DOLCE	$\mathcal{SHIF}$	351	2

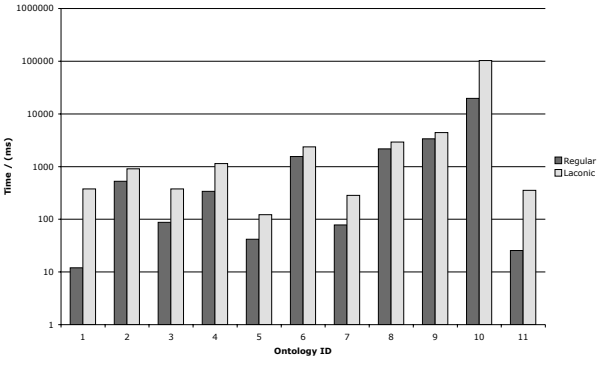

Fig. 1. Times to compute justifications

Figure 1 displays the times for computing regular justifications and laconic justifications. It is clear to see that computing laconic justifications takes longer than computing regular justifications. This is to be expected since the computation of regular justifications is used as a first step for computing laconic justifications. Nevertheless, the mean time for computing laconic justifications is acceptable for the purposes of computing laconic justifications on demand in an ontology development environment. It can be seen from Figure 1 that Ontology 10, which is the Chemical ontology, required the most time computationally—the average time was 100000 milliseconds (1 min 40 seconds) per entailment. Indeed, for the Chemical ontology, the average time per entailment to compute regular justifications is 20 seconds. The reason for this longer computation time, is that on average, each entailment in the chemical ontology has 9 justifications, with one entailment topping 26 justifications.

8 Observations on Computed Laconic Justifications

The laconic justifications that were computed in the experiment, and the relationship that these have with their corresponding regular justifications, exhibited

several properties that verify the motivational reasons for laconic justifications (Section 3). Examples of masking and larger numbers of regular justifications than laconic justifications for a given entailment were present. This section discusses these examples.

8.1 Masking

As described in Section 3 one of the main issues with regular justifications is that for a given entailment they can *mask* other justifications. An example of such masking occurs in the DOLCE ontology. The entailment $\texttt{quale} \sqsubseteq \texttt{region}$ has a single justification: $\{\texttt{quale} \equiv \texttt{region} \sqcap \exists\ \texttt{atomic-part-of.region}\}$. However, computing laconic justifications for this entailment reveals that there are further justifications that are masked by this regular justification. There are three laconic justifications, the first being $\{\texttt{quale} \sqsubseteq \texttt{region}\}$, which is directly obtained as a weaker form of the regular justification. This first laconic justification could be identified in Swoop using the strike out feature (The conjunct $\exists\ \texttt{atomic-part-of.region}$ would be struck out). More interestingly, there are two additional laconic justifications: $\{\texttt{quale} \sqsubseteq \exists \texttt{atomicPartOf.region}, \texttt{atomicPartOf} \sqsubseteq \texttt{partOf}, \texttt{partOf} \sqsubseteq \texttt{part}^{-}, \texttt{region} \sqsubseteq \forall \texttt{part.region}\}$ and also $\{\texttt{quale} \sqsubseteq \texttt{atomicPartOf.region}, \texttt{atomicPartOf} \sqsubseteq \texttt{atomicPart}^{-}, \texttt{atomicPart} \sqsubseteq \texttt{part}, \texttt{region} \sqsubseteq \forall \texttt{part.region}\}$

Masking is surprising in general, and the above example is a nice illustration of how this information would not be revealed with regular justifications. In such cases the user or developer of an ontology would be completely unaware of these further justifications when attempting to formulate a repair strategy or when simply trying to gain a deeper understanding of the ontology.

8.2 Number of Justifications Versus Number of Laconic Justifications

Figure 2 displays the mean number of justifications per entailment. For ontologies 1, 2, 5, 6 and 9 (five out of eleven ontologies) the number of laconic justifications coincides with the number of regular justifications. However, for ontologies 3, 4, 7, 8 and 11, the mean number of laconic justifications per entailment is greater than that of regular justifications. This is an indication that internal or external masking is occurring for a significant number of ontologies, corresponding to the second and third motivations in Section 3. Again, these extra justifications would not be salient to a user who only works with regular justifications, and would mean that it might be impossible to gain a full understanding of an ontology when devising a repair plan.

In ontology 10, the Chemical ontology, it is evident that the mean number of laconic justifications per entailment is *less* than the number of regular justifications. In fact, in this particular ontology, there is an entailment with *six* regular justifications and only *two* laconic justifications. This situation also occurs in places in the PeriodicTableComplex ontology, where there are a large number of entailments that have two regular justifications and one laconic justification. In

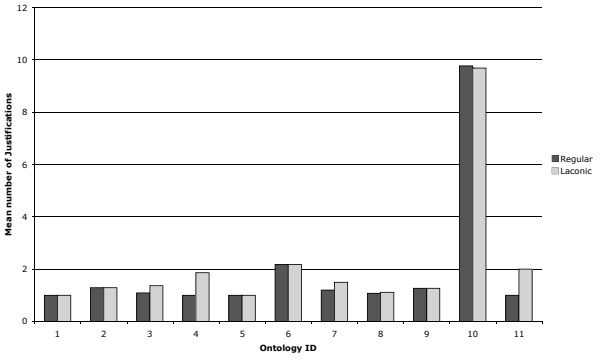

Fig. 2. Number of regular justifications versus the number of laconic justifications

the case where there are fewer laconic justifications than regular justifications, the laconic justifications highlight a common core amongst the regular justifications that is responsible for the entailment. Working with this information when repairing an ontology potentially minimises the possibility of applying an "over-repair" that could occur by quasi-independently examining each regular justification.

9 Exploiting Laconic and Precise Justifications

It should be noted that the work presented in this paper has not covered how laconic, or in particular how precise justifications, should be presented to users, incorporated into workflows or how they might be exploited in various reasoning services. While it is easy to imagine how they can be used to provide an enhanced and complete service for striking out irrelevant parts of axioms, they could also be used as a basis for measuring incoherence, measuring complexity of understanding, as metrics for repair services, and in ontology refactoring and simplification services. The issue of presenting laconic and precise justifications and incorporating them into various services is the topic of future work.

10 Conclusions and Future Work

This paper has presented a formal definition of fine-grained justifications in the form of laconic justifications and precise justifications. The definition of laconic justifications captures the intuitive notion of fine-grained justifications from previous related work, while the definition of precise justifications captures the notion of being able to generate a semantically minimal repair. An optimised algorithm to computed laconic justifications in accordance with this definition has been given and it has been shown that it is feasible to compute laconic justifications (and hence precise justifications) in practice. The definition and evaluation has provided a deeper insight into the properties of laconic justifications and

how laconic justifications and their precise counterparts might eventually be exploited. Finally, it should be noted that the definition that has been presented in this paper assumes a consistent ontology. Dealing with inconsistent ontologies is the subject of future work.

References

1. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. PhD thesis, The Graduate School of the University of Maryland (2006)
2. Kalyanpur, A., Parsia, B., Hendler, J.: A tool for working with web ontologies. International Journal on Semantic Web and Information Systems 1 (2005)
3. Horridge, M., Tsarkov, D., Redmond, T.: Supporting early adoption of owl 1.1 with protégé-owl and fact++. In: OWL: Experiences and Directions (2006)
4. Lam, S.C.J.: Methods for Resolving Inconsistencie. In Ontologies. PhD thesis, Department of Computer Science, Aberdeen (2007)
5. Kalyanpur, A., Parsia, B., Grau, B.C.: Beyond asserted axioms: Fine-grain justifications for OWL-DL entailments. In: Proc. of DL (2006)
6. 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 1(1), 7–26 (2003)
7. Baader, F., Hollunder, B.: Embedding defaults into terminological representation systems. J. of Automated Reasoning 14, 149–180 (1995)
8. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proc. of IJCAI (2003)
9. Meyer, T., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In: Proc. of AAAI (2006)
10. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. In: Proc. of KR 1992, pp. 306–317. Morgan Kaufmann, San Francisco (1992)
11. Deng, X., Haarslev, V., Shiri, N.: Measuring inconsistencies in ontologies. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, Springer, Heidelberg (2007)
12. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic el. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) KI 2007. LNCS (LNAI), vol. 4667, pp. 52–67. Springer, Heidelberg (2007)
13. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. Journal of Symbolic Computation (1986)
14. Motik, B., Shearer, R., Horrocks, I.: Optimized reasoning in description logics using hypertableaux. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 67–83. Springer, Heidelberg (2007)
15. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics 5(2) (2007)