Undefined 1 (2009) 1–5
IOS Press

The OWL Explanation Workbench: A toolkit for working with justifications for entailments in OWL ontologies

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Matthew Horridge [a,*], Bijan Parsia [b] and Ulrike Sattler [b]
[a] *Stanford Center for Biomedical Informatics Research, Stanford University, California, USA*
E-mail: matthew.horridge@stanford.edu
[b] *School of Computer Science, The University of Manchester, Oxford Road, Manchester, United Kingdom*
E-mail: {parsia,sattler}@manchester.ac.uk

Abstract. In this article we present the Explanation Workbench, a library and tool for working with justification-based explanations of entailments in OWL ontologies. The workbench comprises a software library and Protégé plugin. The library can be used in standalone OWL API based applications that require the ability to generate and consume justifications. The Protégé plugin, which is underpinned by the library, can be used by end-users of Protégé for explaining entailments in their ontologies. Both the library and the Protégé plugin are open-source software and are available for free on GitHub.

Keywords: Justifications, Explanation, OWL, Ontologies

1. Introduction

An ontology is a machine processable artefact that captures the relationships between concepts and objects in some domain of interest. In 2004 the Web Ontology Language, OWL [37,56,25], became a World Wide Web Consortium Standard. Since then, it has become the most widely used ontology language, being adopted all over the world by academia and industry alike.

One of the key aspects of OWL is that it is built upon the foundations of a *Description Logic*. Description logics [3] are a family of knowledge representation languages which are typically *decidable fragments* of First Order Logic. This logical foundation gives statements made in OWL a precisely defined, unambiguous meaning. Moreover, it makes it possible to specify various automated reasoning tasks, and to use "off the shelf" description logic reasoners such as FaCT++ [72], HermiT [60], Pellet [68], Racer [26], ELK [46] and CEL [1] for computing relationships between the various concepts and objects that are expressed in an ontology. With reasoning, conclusions that were implicit, but not stated in the original ontology, can be made explicit—conclusions can be *inferred*.

A consequence of the fact that OWL corresponds to a highly expressive description logic is that unexpected and undesirable inferences (entailments), can arise during the construction of an ontology. The reasons as to why an entailment holds in an ontology can range from simple localised reasons through to highly non-obvious reasons. In the case of very large ontologies such as the National Cancer Institute The-

[*] Corresponding author

saurus [22], which contains over 80,000 axioms, or the large medical ontology SNOMED [69], which contains over 300,000 axioms, manually pinpointing the reasons for an entailment can be a wretched and error prone task. Without automated explanation support it can be very difficult to track down the axioms in an ontology that give rise to entailments. It is for this reason that automated explanation is an important topic in this area.

Indeed, since OWL became a standard, there has been a widespread demand from ontology developers for tools that can provide explanations for entailments. Some common tasks that prompt a demand for explanation facilities are:

1. **Understanding entailments**—A user browsing an ontology notices an entailment and opportunistically decides to obtain an explanation for the entailment in order to find out what has been stated in the ontology that causes the entailment to hold.
2. **Debugging and repair of ontologies**—A user is faced with an incoherent or inconsistent ontology, or an ontology that contains some other kind of undesirable entailment. They need to determine the causes of the entailment in order to understand why it holds so that they can generate a repair plan.
3. **Ontology comprehension**—A user is faced with an ontology that they have not seen before. In order to get a better picture of the ontology they use various metrics such as the number of entailments, the average number of explanations for an entailment and so on. This helps them to build up an image of how complex the ontology is in terms of expressivity. Is also provides them with more information if they need to decide whether they like the ontology or not.

1.1. *Justification Based Explanation*

In the world of OWL there has been a significant amount of research devoted to the area of explanation and ontology debugging. In particular, research has focused on a specific type of explanation called *justifications* [4,65,39,5]. A justification for an entailment in an ontology is a minimal subset of the ontology that is sufficient for the entailment to hold. The set of axioms corresponding to the justification is minimal in the sense that if an axiom is removed from the set, the remaining axioms no longer support the entailment. More precisely, given an ontology $\mathcal{O}$ and an entailment η such that $\mathcal{O} \models \eta$, $\mathcal{J}$ is a justification for η in $\mathcal{O}$ if $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \eta$, and for all $\mathcal{J}' \subsetneq \mathcal{J}$ it is the case that $\mathcal{J}' \not\models \eta$.

Justifications have turned out to be a very attractive form of explanation: They are conceptually simple, they have a clear relationship with the ontology from which they are derived, there are off-the-shelf algorithms for computing them, and there are simple presentation strategies which work well most of the time.

In this article we present the Explanation Workbench, a library and tool for working with justification-based explanations of entailments in OWL ontologies. The workbench comprises a software library and Protégé plugin. The library can be used in standalone OWL API based applications that require the ability to generate and consume justifications. The Protégé plugin, which is underpinned by the library, can be used by end-users of Protégé for explaining entailments in their ontologies. Both the library and the Protégé plugin are open-source software and are available for free on GitHub.

2. Preliminaries

OWL, Ontologies and Entailments OWL is the latest standard in ontology languages from the World Wide Web Consortium. Rather than providing a detailed description of the language, we refer the interested reader to the OWL primer [29], the OWL overview [25] and associated documents, and we simply provide the bare minimum details required for this article. An OWL ontology is a finite set of *axioms* (statements) that describe the domain modelled by the ontology. Axioms may be divided into logical axioms (corresponding to sentences in the logic that underpins OWL) and non-logical axioms (used for labelling and annotating things). From this point forward we ignore non-logical axioms and when we say axioms we mean logical axioms. We say that axioms contained within an ontology are *asserted axioms*—that is, they have been explicitly written down, or asserted, by a person (or process). The asserted axioms in an ontology give rise to *entailed axioms*, or simply *entailments*, which follow as a logical consequence of the asserted axioms. For example, the ontology $\mathcal{O}$ = { Golf SubClassOf Car, Car SubClassOf Vehicle, Car SubClassOf hasPart some Engine } contains three asserted axioms and, amongst other things, entails the (non-asserted) axiom { Golf

SubClassOf Vehicle }[1]. For a given ontology $\mathcal{O}$, and axiom α we write $\mathcal{O} \models \alpha$ if $\mathcal{O}$ entails α. Further more, if $\mathcal{O} \models \alpha$ there is a (possibly empty) subset of $\mathcal{O}$ that causes it to entail α. Finally, entailments can automatically be checked using an OWL *reasoner*.

Justifications A justification $\mathcal{J}$ for an entailment α in an ontology $\mathcal{O}$ is a minimal subset of $\mathcal{O}$ that is sufficient to entail α. More precisely, $\mathcal{J}$ is a justification for $\mathcal{O} \models \alpha$ if $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \alpha$, and for any $\mathcal{J}' \subsetneq \mathcal{J}$ it is the case that $\mathcal{J}' \not\models \alpha$.

Laconic Justifications Justifications operate at the level of asserted axioms. While it is the case that all axioms in a justification are required for the entailment in question to hold, it may be the case that not all *parts* of axioms within a justification are required to support that entailment. For example, given $\mathcal{J}$ = { Golf SubClassOf Car and hasManufacturer value Wolkswagen } for the entailment Golf SubClassOf Car, it is clear that the conjunct hasManufacturer value Wolkswagen is superfluous as far as the entailment is concerned. Intuitively, a justification, none of whose axioms contain superfluous parts for the entailment in question, is known as a *laconic justification*. A more precise definition and characterisation of laconic justifications may be found in [30].

Computing Justifications Algorithms for computing justifications may be categorised into glass-box algorithms and black-box algorithms. The distinction between these two categories is down to the part played by a reasoner when computing justifications. A glass-box algorithm implementation is tightly interwoven with a *specific* reasoner and computes justifications during standard reasoning tasks (during entailment checking for example) as a side effect of reasoning. A black-box algorithm, on the other hand, is not tied to a specific reasoner, but can be used with any OWL reasoner. Black-box implementations simply use a reasoner as an entailment checking oracle when computing justifications. In the early days of explanation research, it was thought that glass-box algorithms were essential for good performance. However, recent research has shown that this is not the case, and that black-box algorithms can be used in applications that require robust, high performance justification computation subroutines [30]. In this work, the explanation workbench and API supports either glass-box or black-box implementations, however the reference implementation is a black-box algorithm.

[1]Note that we use a variant of the Manchester Syntax for writing axioms as this is the syntax used by Protégé and in the screenshots throughout this article.

3. The Rise to Prominence of Justifications

In order to provide some context for this work and our framework, in what follows we provide an overview of justification based explanation, tracing it back to to its roots and forward to its use today.

Explanation has long been recognised as a key component in knowledge representation systems [55,9,36, 53]. One of the most prominent early Description Logic systems to feature an explanation component was the CLASSIC [11] system, where explanation was recognised as being very important for the usability of the system [55]. In this early work, an explanation was essentially regarded as a proof, or a fragment of a proof, which explained how a reasoner proved that an entailment held in some ontology. In fact, there was a general feeling that an explanation system had to be closely allied with the reasoning system that proved entailments [54], and that proof based explanations were essentially *declarative* views on the structural reasoning procedures that were used at the time. This was a point of view that was maintained when more expressive Description Logics, such as $\mathcal{ALC}$ [66], which featured sound and complete tableau-based reasoning procedures [6], started to come to the fore. Indeed, the notion of proof-based explanations was defined for $\mathcal{ALC}$ in [10], extended and implemented in one form or another in [49] and [52], and also relatively recently adapted to the DL-Lite [13] family of Description Logics in [12], and the $\mathcal{EL}++$ [2] based OWL2EL profile [59] in [45].

3.1. From Proofs to Justifications

While the ideas of proof based explanations in Description Logics are still around, the years between 2003 and 2005 marked a turning point in explanation for OWL based systems. Specifically, the fundamental idea of what constituted an explanation completely changed. This paradigm shift was centred around two seminal pieces of work: The first by Schlobach and Cornet [65] in 2003, and the second by Parsia, Kalyanpur et al. [61] in 2005.

In [65], Schlobach and Cornet presented work on diagnosing and repairing ontologies that contained unsatisfiable concepts. Their work, part of which turned out to be closely related to early work by Baader and Hollunder [4], was motivated by the DICE ontology [17,18] which is a large Description Logic based ontology for intensive care. Most importantly, Schlobach and Cornet introduced the notion of *Minimal Unsatisfiable Preserving Sub-TBoxes* (MUPS). These are minimal subsets of an ontology that are sufficient for entailing the unsatisfiability of a particular concept, and in essence are justifications for unsatisfiable classes.

In 2004, the Web Ontology Language OWL became a W3C recommendation (standard). However, people had already begun to build OWL ontologies before this. First using early editors such as OilEd [9], which were originally built for editing DAML+OIL [35] ontologies—a precursor to OWL, and then using early versions of Protégé [47] and Swoop [40]. During this time, despite earlier work on proof based explanation, it became apparent that the tools and techniques for debugging inconsistent ontologies, or ontologies containing unsatisfiable classes, were practically non-existent. Ontology developers used trial and error to resolve problems, essentially ripping out axioms from their ontologies until classes turned satisfiable or the ontologies turned consistent. The only indications and debugging cues that were available were error messages saying that an ontology was inconsistent, or class names were painted in red to indicate that classes were unsatisfiable. It was obvious to those in the area that some sort of automated debugging support was desperately needed.

It was at this time that Parsia and Kalyanpur began to investigate techniques for the debugging and repair of OWL ontologies. In [61], and then in subsequent publications [44,42,43], they introduced various important OWL ontology debugging techniques. Ultimately, Kalyanpur and Parsia were responsible for seeing the value of justifications as explanation and debugging devices and bringing justification based explanation to the masses in Swoop. This work culminated in Kalyanpur's PhD thesis [39] which brought the notions of justifications, root unsatisfiable classes[2], and justification based repair together and demonstrated the overwhelming benefit of justification-based debugging support for repairing ontologies.

[2]A root unsatisfiable class is a concept name, in the signature of an ontology, whose unsatisfiability does not depend upon the unsatisfiability of some other concept name in the signature of that ontology

3.2. *Justification Based Explanation Today*

In years following both Schlobach's and then Parsia and Kalyanpur's work there has been a huge interest in this area, and other researchers began to work on methods and techniques for computing and working with justifications [51,57,48,5,7,41,50]. Such was the demand for explanation by people developing and working with ontologies, many of the major ontology development environments began to offer justifications as a popular form of explanation.

While the primary use of justifications is still explanation, people are also increasingly using them to get a feel of the "shape" of an ontology. In this case, when people come across an arbitrary entailment, they request a justification for the entailment so as to get a feel as to what kinds of axioms and constructs in the ontology give rise to the entailment.

3.3. *Justifications in Auxiliary Services*

Justifications are also increasingly being used for purposes other than explanation. For example, they are used for enriching ontologies [64], belief base revision [27], scalable ABox reasoning [19], incremental reasoning [15], reasoner completeness testing [71], meta-modelling support [21], default reasoning [63] and elimination of redundant axioms in ontologies [24]. There is plenty of evidence that they have utility within the OWL and Description Logic communities beyond the starting point of explanation.

3.4. *Justifications in Other Fields*

Finally, although Schlobach and Cornet were the first to introduce minimal entailing sets of axioms as forms of explanation to the Description Logic community, the same notion is also used in other communities. For example, *minimal conflict sets* (MCSs) in the area of model based diagnosis [62,28] correspond to justifications. Similarly, *irreducible inconsistent systems* (IISs) [14] in the area of linear programming also correspond to justifications, and kernels in belief revision. Lastly, the Propositional Logic reasoning community use Minimal Unsatisfiable Sub-formulae (MUSes) for explaining why a set of clauses is unsatisfiable [8].

3.5. *Summary*

In summary, justifications are a widely used form of explanation. They dominate the current crop of tools and techniques for debugging and repairing ontologies, and they are widely used for purposes other than explanation.

The widespread use, and potential utility, of justifications underlines the importance of having a robust off-the-shelf, tool independent, API and reference implementation for computing justifications. Furthermore, an end-user facing tool that uses such an API for computing justifications is likely to be of benefit for users of a mainstream ontology editor such as Protégé.

4. The OWL Explanation Workbench

The OWL Explanation Workbench is a software suite for computing, analysing and viewing justifications for entailments in OWL ontologies. The workbench comes in two distinct parts: (1) A standalone Java library for computing and examining justifications, and (2) A graphical user interface, in the form of a Protégé plugin, for computing and viewing justifications for entailments within the Protégé ontology development environment. The latter uses the former as a dependency under the hood.

4.1. The OWL Explanation Library

Using the Library The OWL Explanation Library is written in Java and interfaces with the OWL API [32]. It is open source and is available on *GitHub*.[3] The library is set up as a Maven project and releases are published on Maven Central[4] so that any Maven project may simply import and use the library as a dependency. Detailed instructions for using the library may be found on the library's GitHub home page.

An API and Reference Implementation The library provides an explanation Application Programmer Interface (API) with high-level interfaces for representing justifications and strategies for computing justifications. It also contains an optimised reference implementation that includes black-box algorithms for computing both regular and laconic justifications for an entailment in a set of axioms (or OWL API `OWLOntology` object). It is possible to compute either a *single* justification for a given entailment, or, incrementally compute *all* justifications for an entailment.

While the out-of-the-box justification finding algorithms are black-box algorithms, the use of Java interfaces to represent justification computation strategies allows any concrete justification generator, including glass-box based algorithms, to be "plugged in" to the framework—clients that use these interfaces can therefore remain oblivious to changes in the concrete implementation that is used to find justifications.

The library uses a suite of optimisations, including: syntactic-locality-based modules [16] (to limit search space and boost entailment checking performance), a structural expansion stage [39] (which also helps limit search space), a divide and conquer pruning strategy [67] (which has been empirically shown to offer best performance over similar strategies), and standard Hitting Set Tree (HST) optimisations [62,20] (for computing all justifications). Together these optimisations have been empirically found to result in good performance for naturally occurring ontologies [30,38].

Utilities In naturally occurring ontologies there can be numerous justifications per entailment. In particular, there can be hundreds of justifications for an inconsistent ontology (the entailment owl:Thing SubClassOf owl:Nothing) [30,34]. The API therefore features the ability to monitor the progress of computing justifications. This simple, but handy, functionality allows applications the freedom to terminate the computation run if they have found enough justifications for their purpose. In tools such as the explanation workbench plugin for Protégé, the practical upshot of this is that it allows a progress bar to be displayed so that users can choose to cancel the computation at their discretion. The API also allows a timeout to be specified so that clients can simply compute as many justifications as possible within some time period.

Finally, the explanation workbench library contains a heuristic implementation for computing root unsatisfiable classes [39] contained in the signature of an incoherent ontology[5]. Intuitively, a derived unsatisfiable class is an unsatisfiable class whose (un)satisfiability depends upon the (un)satisfiability of another class. For example, if A is unsatisfiable, given the axiom B SubClassOf hasPart some A we say that B is a de-

[3] `https://github.com/matthewhorridge/owlexplanation`

[4] `http://maven.org` with a groupId of `net.sourceforge.owlapi` and an artifactId of `owlexplanation`

[5] An unsatisfiable class is a class that is always interpreted as the empty set. Thus, it cannot have any instances. Unsatisfiable classes are generally the result of modelling errors and should be fixed before the ontology in questions is published.

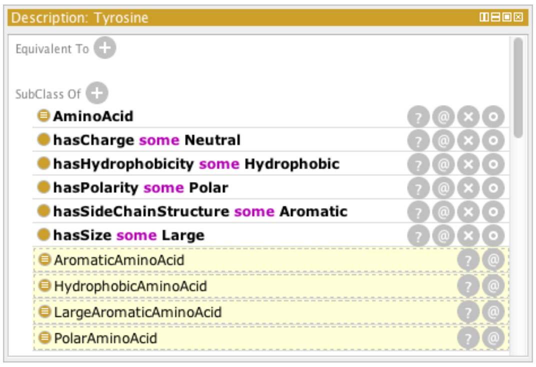

Fig. 1. The Explain Inference Buttons. Clicking on a question mark button will cause justifications to be computed for the axioms underlying the corresponding row. For example, clicking the button next to AliphaticAminoAcid causes justifications for Tyrosine SubClassOf AliphaticAminoAcid to be computed. Note that the rows with the yellow background indicated inferred axioms that are not asserted. Rows with the white background indicate axioms that are asserted. It is possible to ask for justifications for both, since an asserted axiom may also be entailed due to some reasoner besides being asserted.

rived unsatisfiable class. Classes that are not derived-unsatisfiable classes are known as root-unsatisfiable classes. The ability to find root-unsatisfiable classes is useful because computing justifications for root-unsatisfiable classes, and then using these justifications to repair the root-unsatisfiable classes automatically results in derived-unsatisfiable classes being repaired. This functionality was used to great effect in an OWL version of the TAMBIS ontology [70],[6] which has one hundred and forty four unsatisfiable classes in its signature with just three of these classes being root-unsatisfiable classes.

4.2. The OWL Explanation Workbench Protégé Plugin

The other major part of the explanation workbench is a user-facing plugin for Protégé. The plugin allows justifications for inferences that are shown in various places in the Protégé user-interface to be computed. The plugin is built upon the afore mentioned library and therefore supports both the computation of regular justifications and laconic (fine-grained) justifications for entailments.

Invoking and Using the Workbench In order to invoke the workbench from within Protégé, the user presses one of the "Explain inference" buttons that are visible through the Protégé user-interface after reasoning has been invoked. An example, taken from the Class Description View, is shown in Figure 1—the explain inference buttons are the circles containing question marks. Upon pressing the button computation of regular justifications for the selected entailment begins. The reference implementation black-box justifications finding algorithm is used in conjunction with the reasoner that is selected in Protégé as a backing-reasoner (i.e. if FaCT++ is selected and was used to perform reasoning in Protégé, then FaCT++ will also be used for computing justifications). Furthermore, justifications can be computed for both consistent *and inconsistent* ontologies. Prior to the Explanation Workbench Library and Plugin being released, an off-the-shelf solution for computing justifications for inconsistent ontologies was not available.

During the computation phase, a progress window is displayed that indicates how many justifications for the given entailment have so far been computed. Up until the point where all justifications have been computed, the user is free to halt the process so that they can see the justifications that have so far been computed. This feature was found to be necessary as certain entailments in naturally occurring ontologies can have hundreds of justifications [30]. While it is usually possible to compute all justifications for an entailment [30], in some cases it is not possible to compute all justifications within a reasonable time-frame (e.g. 10 minutes). In these cases, it is typically still extremely beneficial to examine the justifications that have been computed, as this allows the user to get a feeling for why an entailment holds and can also allow them to incrementally repair an ontology [34].

At the point when all justifications have been computed, or the user halted the current run, the explanation workbench window shown in Figure 2 is displayed. The window shows the computed justifications in a series of boxes, with one box for each justification.

The ordering of justifications within the workbench window is computed using a notion syntactic complexity. Justifications with fewer types of class expressions and axioms are displayed towards the head of the list, while more complex justifications that contain a larger number of axiom and class expression types are displayed towards the tail of the list. While this strategy needs empirically verifying in order to determine whether or not it is truly beneficial to the user, the intuition behind it is that simpler justifications, that allow users to quickly spot errors, or determine why an entailment holds, are shown first.

[6] http://robertdavidstevens.wordpress.com/2010/12/18/the-tambis-ontology/

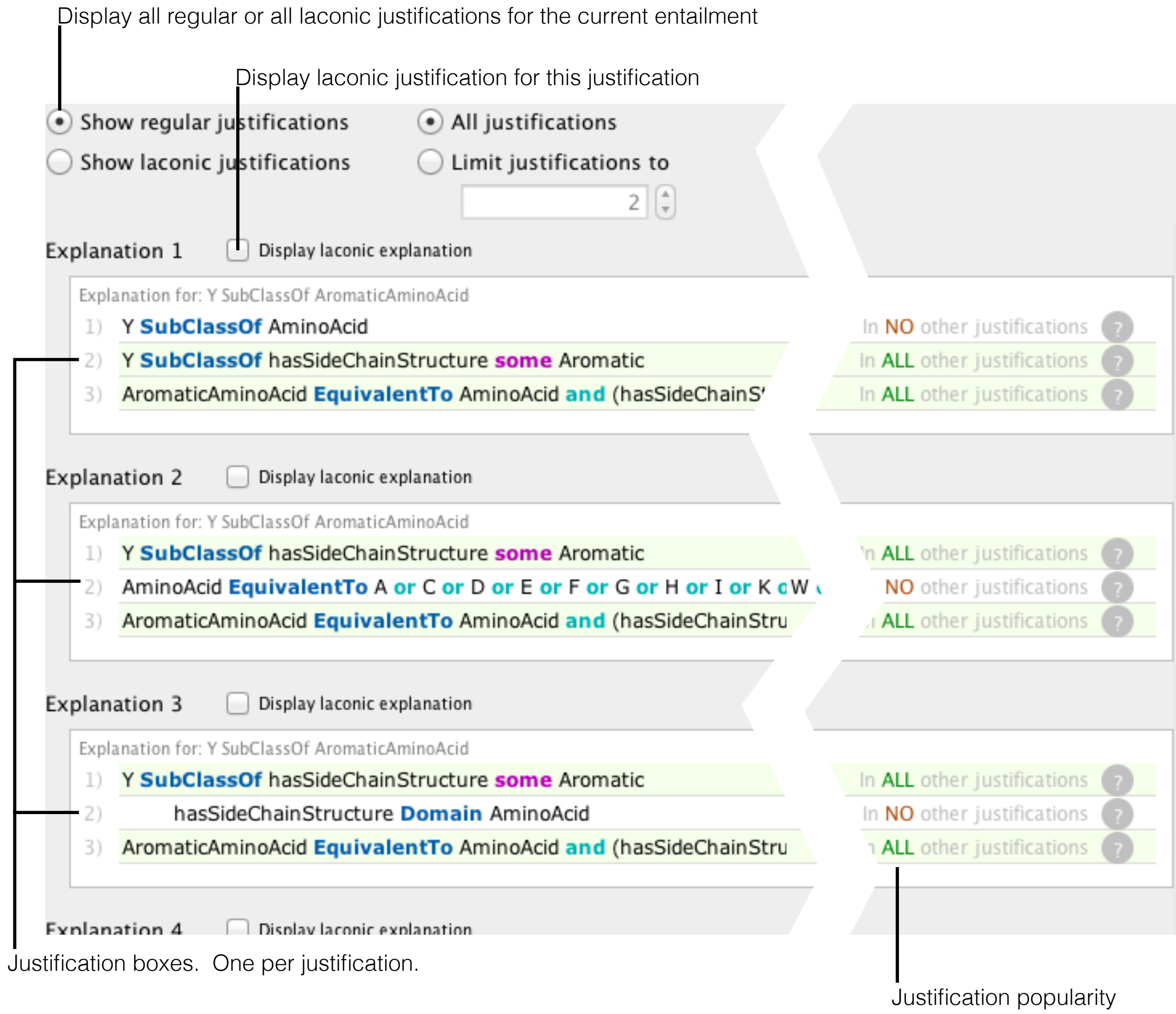

Fig. 2. The Justification Workbench Window

Justification Presentation Each axiom within a justification is shown on a separate line (Figure 3) and is numbered for ease of reference in publications or presentations. Axioms are displayed using the Manchester Syntax [33], which is used throughout Protégé as the default syntax for viewing and editing class expressions. Furthermore, the class, property and individual names that appear in a justification are hyperlinked and users may click on them to select the entity in the main editor window of Protégé. This allows the user to examine other parts of an entity definition that may not appear in a justification, but which may be useful for deciding upon a repair strategy.

Ordering and Indentation A simple ordering and indenting strategy, that was first proposed by Kalyanpur [39], is used for displaying the axioms in a justification. In essence the strategy determines the position of an axiom in the list based on its signature and the positions of entities within the axiom itself. Roughly speaking, axioms that define entities used in some other axiom will appear indented below that axiom. For example, taking A SubClassOf B and C and B SubClassOf D, the second axiom will appear indented after the first, because the first axiom uses B to define A and the second axiom defines B. An example of the effect of indentation is shown in Figure 3, which presents (part of) a justification for an entailment from from the clinical ontology SNOMED-CT (courtesy [58]).

Highlighting and Popularity As can be seen from Figure 2, axioms can be painted with a green or white

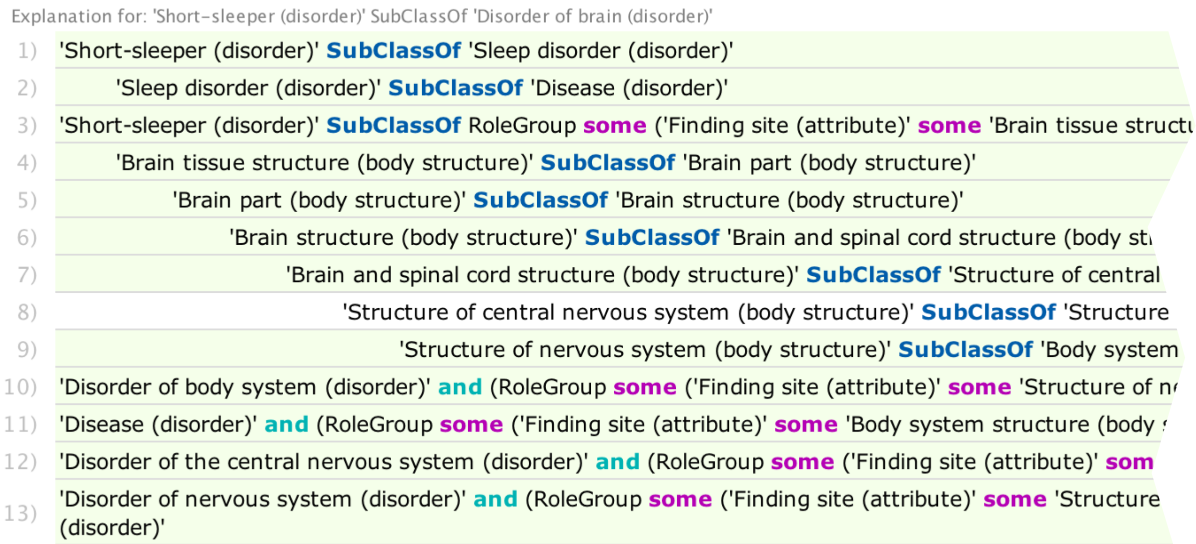

Fig. 3. An example of this indentation and ordering of axioms that is used to aide reading. This example is taken from an explanation for an entailment 'Short-sleep (disorder)' SubClassOf 'Disorder of brain (disorder)' in SNOMED-CT. The second axiom, which states that Sleep disorder (disorder) is a kind of Disease (disorder) is indented and placed below the first axiom, which uses Sleep disorder (disorder) as part of its definition.

background. A green background indicates that an axiom appears in all other justifications that are shown in the workbench window. A white background indicates that this is not the case. For any given axiom, the number of justifications that it appears in (called *popularity* here) is displayed to its right. These metrics can be useful when trying to understand the importance, or impact, of an axiom on an entailment. They can also be used to derive a mechanical repair: To remove an entailment from an ontology, one axiom from each justification must be removed from the ontology; removing axioms that appear in a larger number of justifications can result in fewer axioms (and hopefully less entailments) being removed from the original ontology.

Finally, selecting an axiom in one justification causes it to be highlighted in any other justifications that it appears in.

Displaying Laconic Justifications Recalling that a laconic justification is a justification whose axioms don't contain any superfluous parts, the workbench window features controls for displaying a *laconic version* of a given justification, or for displaying *all laconic justifications* for an entailment (Figure 2).

In the first case, selecting the "Show laconic explanation" checkbox above a given justification will toggle the display of that justification with a laconic version of it. This can be useful when examining a specific justification as it will cut out the superfluous parts, that can be distracting [31], from that justification.

Selecting the "Show laconic justifications" radio button in the workbench window will compute and display all laconic justifications for the given entailment. It is important to realise that there can be a smaller or a larger number of laconic justifications for an entailment when compared to the number of regular justifications. This reflects the fact that *masking* can take place—regular justifications can obscure multiple reasons for an entailment (via *internal* and *external* masking [30]), and they can also make it appear that there are more reasons for a given entailment than is truly the case (via *shared-core* masking [30]). Although the ability to toggle between a regular and laconic justification exposes the utility of laconic justifications, we do intend to improve the presentation here by adding a strikeout based presentation (similar to the heuristic based strikeout that may be found in the ontology editor Swoop). We also intend to add support for determining the asserted axioms that are the source of axioms in a laconic justification.

5. Uptake and Usage

The OWL Explanation Workbench plugin for Protégé is bundled with and distributed along with Protégé 5. While we do not have precise metrics for how often the plugin is actually used, we note that Protégé has over 100,000 registered users. Furthermore, the plugin is tightly integrated with the main editor and appears to users by default, without any extra configuration on their part.

6. Conclusions and Future Work

In this article we have presented the explanation workbench a Java library for computing justifications that interfaces with the OWL API and a user interface in the form of a Protégé plugin. The library and plugin are freely available under an open source license, along with all source, code on Github. The explanation library can be used in stand alone applications that require the ability to compute and consume justifications or laconic justifications, and the explanation workbench plugin is bundled with the Protégé 5 distribution.

References

[1] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006.

[2] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In eslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 364–369. Professional Book Center, August 2005.

[3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, D Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[4] Franz Baader and Bernhard Hollunder. Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning*, 14(1):149–180, 1995.

[5] Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}$. In Joachim Hertzberg, Michael Beetz, and Roman Englert, editors, *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI, KI 2007, Osnabrück, Germany*, volume 4667 of *Lecture Notes in Computer Science*, pages 52–67. Springer, September 2007.

[6] Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.

[7] Franz Baader and Boontawee Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *Proceedings of the 3rd Knowledge Representation in Medicine Conference (KR-MED'08): Representing and Sharing Knowledge Using SNOMED*, 2008.

[8] James Bailey and Peter J. Stuckey. Discovery of minimal unsatisfiable subsets of contraints using hitting set dualization. In *Practical Aspects of Declarative Languages (PADL 05)*, 2005.

[9] Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: A Reason-able ontology editor for the semantic web. In *Proc. of the Joint German/Austrian Conf. on Artificial Intelligence (KI 2001)*, number 2174 in Lecture Notes in Artificial Intelligence, pages 396–408. Springer, 2001.

[10] Alex Borgida, Enrico Franconi, and Ian Horrocks. Explaining $\mathcal{ALC}$ subsumption. In Werner Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, August 20-25, 2000*, pages 209–213. IOS Press, 2000.

[11] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: a structural data model for objects. In James Clifford, Bruce Lindsay, and David Maier, editors, *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 58–67. Association for Computing Machinery, Inc. (ACM), June 1989.

[12] Alexander Borgida, Diego Calvanese, and Mariano Rodriguez. Explanation in dl-lite. In *Description Logics*, 2008.

[13] Diego Calvanese, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Dl-lite: Tractable description logics for ontologies. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 602–607, 2005.

[14] John W. Chinneck. Finding a useful subset of constraints for analysis in an infeasible linear program. *Informs Journal on Computing*, 9:164–174, 1997.

[15] Bernardo Cuenca Grau, Christian Halasheck-Wiener, and Yevgeny Kazakov. History matters: Incremental ontology reasoning using modules. In Karl Aberer, Key-Sun Choi, Natalya F. Noy, Guus Schreiber, and Riichiro Mizoguchi, editors, *The Semantic Web - 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 183–196. Springer, 2007.

[16] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Just the right amount: Extracting modules from ontologies. In *WWW 2007, Proceedings of the 16th International World Wide Web Conference, Banff, Canada, May 8-12, 2007*, pages 717–727, 2007.

[17] N. F. de Keizer, Ameen. Abu-Hanna, Ronald Cornet, Johanna H. M. Zwetsloot-Schonk, and Chris P. Stoutenbeek. Analysis and design of an ontology for intensive care diagnoses. *Methods of Information in Medicine*, 38:102–112, 1999.

[18] N. F. de Keizer, F Bakhshi-Raiez, E De Jonge, and Ronald Cornet. Post-coordination in practice: evaluating compositional terminological-system-based registration of icu reasons for admission. *International Journal of Medical Informatics*, 77:828–835, 2008.

[19] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Aaron Kershenbaum, Edith Schonberg, Kavitha Srinivas, and Li Ma. Scalable semantic retrieval through summarization and refinement. In Robert C. Holte and Adele Howe, editors, *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 299–304. AAAI Press, 2007.

[20] Amir Fijany and Farokh Vatan. New approaches for efficient solution of hitting set problem. In *WISICT '04: Proceedings of*

the winter international synposium on Information and communication technologies, pages 1–1. Trinity College Dublin, 2004.

[21] Birte Glimm, Sebastian Rudolph, and Johanna Völker. Integrated metamodeling and diagnosis in OWL 2. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang 0007, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, volume 6496 of *Lecture Notes In Computer Science*, pages 257–272. Springer, 2010.

[22] Jennifer Golbeck, Gilberto Fragoso, Frank Hartel, Jim Hendler, Jim Oberthaler, and Bijan Parsia. National cancer institute's thesaurus and ontology. *Journal of Web Semantics*, 1(1):75–80, December 2003.

[23] Bernardo Cuenca Grau, Pascal Hitzler, Conor Shankey, and Evan Wallace, editors. *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[24] Stephan Grimm and Jens Wissmann. Elimination of redundancy in ontologies. In Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dmitris Plexousakis, Pieter De Leenheer, and Jeff Pan, editors, *The Semantic Web: Research and Applications. 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I*, volume 6643 of *Lecture Notes In Computer Science*, pages 260–274. Springer, June 2011.

[25] OWL Working Group. OWL 2 Web Ontology Language document overview (second edition). Technical report, World Wide Web Consortium, December 2012.

[26] Volker Haarslev and Ralf Möller. RACER system description. In *International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes In Computer Science*, pages 701–705, 2001.

[27] Christian Halaschek-Wiener, Yarden Katz, and Bijan Parsia. Belief base revision for expressive description logics. In Bernardo Cuenca Grau, Pascal Hitzler, Conor Shankey, and Evan Wallace, editors, *Proceedings of the OWLED 06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006.*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[28] Walter Hamscher, Luca Console, and Johan de Kleer, editors. *Readings in Model Based Diagnosis*. Morgan Kaufmann Publishers Inc., June 1992.

[29] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. OWL 2 Web Ontology Language Primer (second edition). Technical report, World Wide Web Consortium, December 2012.

[30] Matthew Horridge. *Justification Based Explanation in Ontologies*. PhD thesis, School of Computer Science, The University of Manchester, 2011.

[31] Matthew Horridge, Samantha Bail, Bijan Parsia, and Uli Sattler. Toward cognitive support for OWL justifications. *Knowledge Based Systems*, 53:66–79, 2013.

[32] Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11–21, February 2011.

[33] Matthew Horridge, Nick Drummond, John Goodwin, Alan L. Rector, Robert Stevens, and Hai Wang. The Manchester OWL syntax. In Grau et al. [23].

[34] Matthew Horridge, Bijan Parsia, Natalya Fridman Noy, and Mark A. Musen. Reasoning based quality assurance of medical ontologies: A case study. In *Proceedings of the American Medical Informatics Annual Symposium (AMIA 2014), November 16th-20th, 2014, Washington DC, USA*, 2014.

[35] Ian Horrocks. DAML+OIL: a reason-able web ontology language. In *Proc. of EDBT 2002*, number 2287 in Lecture Notes in Computer Science, pages 2–13. Springer, March 2002.

[36] Ian Horrocks. Applications of description logics: State of the art and research challenges. In Frithjof Dau, Marie-Laure Mugnier, and Gerd Stumme, editors, *Conceptual Structures: Common Semantics for Sharing Knowledge, 13th International Conference on Conceptual Structures, ICCS 2005, Kassel, Germany, July 17-22, 2005, Proceedings*, volume 3596 of *Lecture Notes In Computer Science*, pages 78–90. Springer, 2005.

[37] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[38] Qiu Ji, Zhiqiang Gao, Zhisheng Huang, and Man Zhu. Measuring effectiveness of ontology debugging systems. *Knowledge-Based Systems*, 71(0):169 – 186, 2014.

[39] Aditya Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD thesis, The Graduate School of the University of Maryland, 2006.

[40] Aditya Kalyanpur, Bijan Parsia, and James Hendler. A tool for working with web ontologies. In *International Journal on Semantic Web and Information Systems*, volume 1, Jan - Mar 2005.

[41] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. Finding all justifications of OWL DL entailments. In Karl Aberer, Key-Sun Choi, Natalya F. Noy, Guus Schreiber, and Riichiro Mizoguchi, editors, *The Semantic Web - 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280. Springer, 2007.

[42] Aditya Kalyanpur, Bijan Parsia, and Evren Sirin. Black box techniques for debugging unsatisfiable concepts. In Ian Horrocks, Ulrike Sattler, and Frank Wolter, editors, *Proceedings of the 2005 International Workshop on Description Logics (DL2005), Edinburgh, Scotland, UK, July 26-28, 2005*, volume 147 of *CEUR Workshop Proceedings*. CEUR-WS.org, July 2005.

[43] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca Grau. Repairing unsatisfiable concepts in OWL ontologies. In *European Semantic Web Conference (ESWC), Budva, Montenegro 2006*, 2006.

[44] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in OWL ontologies. *Journal of Web Semantics*, 3(4), 2005.

[45] Yevgeny Kazakov and Pavel Klinov. Goal-directed tracing of inferences in EL ontologies. In Meghyn Bienvenu, Magdalena Ortiz, Riccardo Rosati, and Mantas Simkus, editors, *Informal Proceedings of the 27th International Workshop on Description Logics, Vienna, Austria, July 17-20, 2014.*, volume 1193 of *CEUR Workshop Proceedings*, pages 221–232. CEUR-WS.org, 2014.

[46] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. The incredible ELK: From polynomial procedures to efficient reasoning with EL ontologies. *Journal of Automated Reason-*

ing, 2013.

[47] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The Protégé OWL plugin: An open development environment for semantic web applications. In Sheila McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *ISWC 04 The International Semantic Web Conference 2004, Hiroshima, Japan*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

[48] Petr Kremen and Zdenek Kouba. Incremental approach to error explanations in ontologies. In *I-KNOW 07. Graz: Graz University of Technology, 2007*, pages 332–339, 2007.

[49] Francis King Hei Kwong. Practical approach to explaining $\mathcal{ALC}$ subsumption. Technical report, The University of Manchester, 2005.

[50] Sik Chun Joey Lam. *Methods for Resolving Inconsistencies In Ontologies*. PhD thesis, Department of Computer Science, Aberdeen, 2007.

[51] Kevin Lee, Thomas Meyer, Jeff Z. Pan, and Richard Booth. Computing maximally satisfiable terminologies for the description logic $\mathcal{ALC}$ with cyclic definitions. In Bijan Parsia, Ulrike Sattler, and David Toman, editors, *Proceedings of the 2006 International Workshop on Description Logics (DL2006), Windermere, Lake District, UK May 30 – June 1, 2006*, volume 189 of *CEUR Workshop Proceedings*. CEUR, June 2006.

[52] Thorsten Liebig and Olaf Noppens. OntoTrack: Combining browsing and editing with reasoning and explaining for OWL Lite ontologies. In Sheila McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *The Semantic Web - ISWC 2004. Third International Semantic Web Conference 2004, Hiroshima, Japan*, volume 3298 of *Lecture Notes in Computer Science*, pages 244–258. Springer, November 2004.

[53] Carsten Lutz, Franz Baader, Enrico Franconi, Domenico Lembo, Ralf Möller, Riccardo Rosati, Ulrike Sattler, Boontawee Suntisrivaraporn, and Sergio Tessaris. Reasoning support for ontology design. In Bernardo Cuenca Grau, Pascal Hitzler, Conor Shankey, and Evan Wallace, editors, *Proceedings of the OWLED 06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, Novemver 10–11, 2006*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, November 2006.

[54] Deborah L. McGuinness. *Explaining Reasoning in Description Logics*. PhD thesis, Rutgers University Department of Computer Science, 1996.

[55] Deborah L. McGuinness and Peter F. Patel-Schneider. Usability issues in knowledge representation systems. In Jack Mostow and Charles Rich, editors, *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 608–614, Menlo Park, CA, USA, July 1998. American Association for Artificial Intelligence.

[56] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium, February 2004.

[57] Thomas Meyer, Kevin Lee, Richard Booth, and Jeff Z. Pan. Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *21st National Conference on Artificial Intelligence, AAAI*, 2006.

[58] J. M. Mortensen, E. P. Minty, M. Januszyk, T. E. Sweeney, A. L. Rector, N. F. Noy, and M. A. Musen. Using the wisdom of the crowds to find critical errors in biomedical ontologies: a study of SNOMED CT. *Journal of American Medical Informatics Association*, Oct 2014.

[59] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language Profiles (second edition). Technical report, W3C – World Wide Web Consortium, 2012.

[60] Boris Motik, Rob Shearer, and Ian Horrocks. Optimized reasoning in description logics using hypertableaux. In *Proc. of the 21st Int. Conf. on Automated Deduction (CADE-21)*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 67–83. Springer, 2007.

[61] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL ontologies. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 20*, pages 633–640. Association for Computing Machinery, Inc. (ACM), May 2005.

[62] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[63] Thomas Scharrenbach, Claudia d'Amato, Nicola Fanizzi, Rolf Grütter, Bettina Waldvogel, and Abraham Bernstein. Default Logics for Plausible Reasoning with Controversial Axioms. In Fernando Bobillo, editor, *Proceedings of the 6th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW-2010), 7th November, 2010, Shanghai, China*, CEUR Workshop Proceedings. CEUR Workshop Proceedings, November 2010.

[64] Stefan Schlobach. Debugging and semantic clarification by pinpointing. In Asunción Gómez-Pérez and Jerome Euzenat, editors, *The Semantic Web: Research and Applications, 2nd European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29–June 1, 2005, Proceedings*, volume 3532 of *Lecture Notes In Computer Science*, pages 226–240. Springer-Verlag Berlin Heidelberg, May 2005.

[65] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 355–362. Morgan Kaufmann, 2003.

[66] Manfred Schmidt-Schauss and Gert Smolka. Attributive concept descreiptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[67] Kostyantyn Shchekotykhin, Gerhard Friedrich, and Dietmar Jannach. On computing minimal conflicts for ontology debugging. In Bernhard Peischl, Neil Snooke, Gerald Steinbauer, and Cees Witteveen, editors, *ECAI 2008 Workshop on Model-Based Systems, July 21-22, Patras, Greece. Affiliated with the 18th European Conference on Artificial Intelligence (ECAI 2008)*, pages 7–11, 2008.

[68] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2), 2007.

[69] Kent A. Spackman and Keith E. Campbell. SNOMED RT: A reference terminology for health care. In Daniel R. Masys, editor, *Proceedings of AMIA Annual Fall Symposium*, pages 640–644, Bethesda, Maryland, USA, October 1997. Hanley and Belfus Inc.

[70] Robert Stevens, Patricia G. Baker, Sean Bechhofer, Gary Ng, Alex Jacoby, Norman W. Paton, Carole A. Goble, and Andy Brass. TAMBIS: transparent access to multiple bioinformatics information sources. *Bioinformatics*, 16(2):184–186, 2000.

[71] Giorgos Stoilos, Bernardo Cuenca Grau, and Ian Horrocks. How incomplete is your semantic web reasoner? In Nestor Rychtyckyj, Daniel Shapiro, Maria Fox, and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, July, 2010 Atlanta, Georgia USA*. AAAI Press, July 2010.

[72] Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.