Extracting Justifications from BioPortal Ontologies

Matthew Horridge[1], Bijan Parsia[2], Ulrike Sattler[2]

[1] Stanford University
California, USA, 94305
{matthew.horridge@stanford.edu}
[2] The University of Manchester
Oxford Road, Manchester, M13 9PL
{bparsia@cs.man.ac.uk}

Abstract. This paper presents an evaluation of state of the art black box justification finding algorithms on the NCBO BioPortal ontology corpus. This corpus represents a set of naturally occurring ontologies that vary greatly in size and expressivity. The results paint a picture of the performance that can be expected when finding all justifications for entailments using black box justification finding techniques. The results also show that many naturally occurring ontologies exhibit a rich justificatory structure, with some ontologies having extremely high numbers of justifications per entailment.

1 Introduction

A justification $\mathcal{J}$ for an entailment η in an ontology $\mathcal{O}$ is a minimal subset of $\mathcal{O}$ that is sufficient to entail η. More precisely, $\mathcal{J}$ is a justification for $\mathcal{O} \models \eta$ (read as $\mathcal{O}$ entails η) if $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \eta$ and for all $\mathcal{J}' \subsetneq \mathcal{J}$ $\mathcal{J}' \not\models \eta$. There can be multiple, possibly overlapping, justifications for a given ontology and entailment. Depending upon context, justifications are also known as MUPS (Minimal Unsatisfiability Preserving Sub-TBoxes) [25] or MINAS (Minimal Axiom Sets) [2].

A *justification finding service* computes justifications for an ontology and an entailment. An implementation of a justification finding service is a key component in many of the *explanation* and *debugging* tools that exist for ontology development environments such as Swoop [17], the RaDON plugin for the NeOn Toolkit [14], the explanation workbench for Protégé-4 [12], the explanation facility in OWL Sight [8], and the explanation view in TopBraid Composer [19]. Justification finding services are also increasingly being used as auxiliary services in other applications for example in incremental reasoning [3], reasoning over very large ABoxes [4], belief base revision [9], meta-modelling support [5], default reasoning [24], eliminating redundant axioms in ontologies [7], and laconic justification finding [13].

Given the prominence and importance of justifications, it is no surprise that there is a large literature on techniques and optimisations for computing them.

Much of this literature [31,30,16,20,18,26,27,26,15] is focused on empirical investigations which, generally speaking, are undertaken to validate specific performance optimisations and implementation techniques. This begs the question as to why further empirical investigation is required. In essence, most of the existing empirical work is performed with *prototype implementations* on small collections of ontologies that are *chosen* to show off the effect of specific optimisations and demonstrate proof of concept. This is obviously a completely valid thing to do. However, many of the experiments do not provide a true picture of how highly optimised and robustly implemented justification finding techniques, that take advantage of all published optimisations, will perform on state of the art ontologies that are now widely available. In addition to this, none of the existing experiments were designed to investigate the justification landscape of a broad corpus of ontologies—that is, there is very little data about the numbers and sizes of justifications that one could encounter in naturally occurring ontologies.

The overall aim of this paper is to therefore present a thorough investigation into the practicalities of computing all justifications for entailments in published, naturally occurring ontologies. In particular, ontologies which are representative of typical modelling and are not tutorial or reasoner test-bed ontologies. The end goal is to provide a view of how modern, robustly implemented and highly optimised justification finding algorithms, coupled with modern highly optimised reasoners, perform on realistic inputs, and to paint a picture of the richness of the justification landscape.

All of the data and software, including ontologies, extracted entailments, entailment test timings, hitting set tree statistics, justifications and other raw results, is available online[3] for third parties to access. It should be of interest to those working in justification based research, ontology comprehension, reasoner development, and module extraction amongst other areas.

2 Justification Finding Techniques

In general, algorithms for computing justifications are described using two axes of classification. The first, the *single-all-axis* is whether an algorithm computes a *single* justification for an entailment or whether it computes *all* justifications for an entailment. The second, the *reasoner-coupling-axis* is whether the algorithm is a *black-box* algorithm or whether it is a *glass-box* algorithm. The categorisation is based entirely on the part played by reasoning during the computation of justifications. In essence, justifications are computed as a *direct consequence of reasoning* in glass-box algorithms, whereas they are *not* computed as a direct consequence of reasoning in black-box algorithms. In this sense, glass-box algorithms are tightly interwoven with reasoning algorithms, whereas black-box algorithms simply use reasoning to compute whether or not an entailment follows from a set of axioms. Because of space constraints, and the fact that black-box justification finding services work with any OWL reasoner, this paper focuses

[3] `http://www.stanford.edu/~horridge/publications/2012/iswc/justextract`

entirely on results for black-box justification finding. A detailed analysis of the differences between glass-box and black-box justification finding algorithm performance may be found in [10]. An advantage of black-box algorithms is that they can be easily and robustly implemented [16]. A perceived disadvantage of black-box algorithms is that they can be inefficient and impractical due to a potentially large search space [29].

Black-Box Algorithms for Computing Single Justifications The basic idea behind a black-box justification finding algorithm is to systematically test different subsets of an ontology in order to find one that corresponds to a justification. Subsets of an ontology are typically explored using an "expand-contract" strategy. In order to compute a justification for $\mathcal{O} \models \eta$, an initial, small, subset $\mathcal{S}$ of $\mathcal{O}$ is selected. The axioms in $\mathcal{S}$ are typically the axioms whose signature has a non-empty intersection with the signature of η, or axioms that "define"[4] terms in the signature of η. A reasoner is then used to check if $\mathcal{S} \models \eta$, and if not, $\mathcal{S}$ is expanded by adding a few more axioms from $\mathcal{O}$. This *incremental expansion* phase continues until $\mathcal{S}$ is large enough so that it entails η. When this happens, either $\mathcal{S}$, or some subset of $\mathcal{S}$, is *guaranteed* to be a justification for η. At this point $\mathcal{S}$ is gradually *contracted* until it is a *minimal* set of axioms that entails η i.e. a justification for η in $\mathcal{O}$.

Black-Box Algorithms for Computing All Justifications When formulating a repair plan for an entailment, or attempting to understand an entailment, it is usually necessary to compute *all* justifications for that entailment. This can be achieved using black-box techniques for finding single justifications in combination with techniques that are borrowed from the field of *model based diagnosis* [1]. Specifically, all justifications can be computed by performing systematic "repairs" of the ontology in question, which eliminate already found justifications, and searching for new justifications after each repair. In order to compute these repairs a classical *a hitting set tree* based algorithm is used. A full discussion of this algorithm, which is based on seminal work by Reiter [23], is beyond the scope of this paper, but suffice it to say, the algorithm is widely used in various fields, has been well used and documented in the field of computing justifications [16], and is reasonably well understood in this area. Due to space constraints a more detailed presentation is not offered here, but a comprehensive overview may be found in Chapter 3 of [10].

3 Materials (The BioPortal Corpus)

The number of published real world ontologies has grown significantly since computing justifications for entailments in OWL ontologies was first investigated from around 2003 onwards. In particular, in the last three years the number of ontologies in the biomedical arena has grown considerably. Many of these ontologies have been made available via the *NCBO BioPortal* ontology repository [22]. At the time of writing, BioPortal provides access to the imports closures of over

[4] For example, the axiom $A \sqsubseteq B$ defines the class name A

250 bio-medical ontologies in various formats, including OWL and OBO[5] [28]. Not only is BioPortal useful for end users who want to share and use biomedical ontologies, it is also useful for ontology tools developers as it provides a corpus of ontologies that is attractive for the purposes of implementation testing. In particular, it provides ontologies that: vary greatly in size; vary greatly in expressivity; are real world ontologies; are developed by a wide range of groups and developers and contain a wide variety of modelling styles; and finally, are not "cherry picked" to show good performance of tools.

Curation Procedure The BioPortal ontology repository was accessed on the 12th March 2011 using the BioPortal RESTful Service API. In total, 261 ontology documents (and their imports closures) were listed as being available. Out of these, there were 125 OWL ontology documents, and 101 OBO ontology documents, giving a total of 226 "OWL compatible" ontology documents that could theoretically be parsed into OWL ontologies.

Parsing and Checking Each listed OWL compatible ontology document was downloaded and parsed by the OWL API. OBO ontology documents were parsed according to the lossless OWL-OBO translation given in [6] and [21]. Any imports statements were recursively dealt with by downloading the document at the imports statement URL and parsing it into the imports closure of the original BioPortal "root" ontology. Each axiom that was parsed into the imports closure was labelled with the name of the ontology document from where it originated. If an imported ontology document could not be accessed (for whatever reason) the import was silently ignored.

Out of the 226 OWL compatible ontology documents that were listed by the BioPortal API, 7 could not be downloaded due to HTTP 500[6] errors, and 1 ontology could not be parsed due to syntax errors. This left a total of 218 OWL and OBO ontology documents that could be downloaded parsed into OWL ontologies. After parsing, four of the ontologies were found to violate the OWL 2 DL global restrictions. In all cases, the violation was caused by the use of transitive (non-simple) properties in cardinality restrictions. These ontologies were discarded and were not processed any further, which left 214 ontologies.

Entailment Extraction Three reasoners were used for entailment extraction: FaCT++, HermiT and Pellet. Each ontology was checked for consistency. Five of the 214 ontologies were found to be inconsistent. Next, each *consistent* ontology was classified and realised in order to extract entailments to be used in the justification finding experiments. Entailed *direct* subsumptions between named classes (i.e. axioms of the form $A \sqsubseteq B$) were extracted, along with *direct* class assertions between named individuals and named classes (i.e. axioms of the form $A(a)$). It was decided that these kinds of entailments should be used for testing purposes because they are the kinds of entailments that are exposed through the user interfaces of tools such as Protégé-4 and other ontology browsers—

[5] OBO may be seen as an additional serialisation syntax for OWL.

[6] An HTTP 500 error is an error code that indicated the web server encountered an internal error that prevented it from fulfilling the client request.

they are therefore the kinds of entailments that users of these tools typically seek justifications (explanations) for. The set of entailments for each ontology was then filtered so that it only contained *non-trivial* entailments in accordance with Definition 1.

Definition 1 (Non-Trivial Entailment). *Given an ontology $\mathcal{O}$, such that $\mathcal{O} \models \alpha$, the entailment α in $\mathcal{O}$ is non-trivial if $\mathcal{O} \setminus \{\alpha\} \models \alpha$*

Intuitively, for an ontology $\mathcal{O}$ and an entailment α such that $\mathcal{O} \models \alpha$, α is a non-trivial entailment in $\mathcal{O}$ either if α is not asserted in $\mathcal{O}$ (i.e. $\alpha \notin \mathcal{O}$) or, α is asserted in $\mathcal{O}$ (i.e. $\alpha \in \mathcal{O}$) but $\mathcal{O} \setminus \{\alpha\} \models \alpha$, i.e. $\mathcal{O}$ with α removed still entails α. In total there were 72 ontologies with non-trivial entailments which accounts for just over one third of the consistent OWL and OBO ontologies contained in BioPortal.

Reasoner Performance Due to practical considerations, a timeout of 30 minutes of CPU time was set for each task of consistency checking, classification and realisation. There were just three ontologies, for which consistency checking (and hence classification and realisation) could not be completed within this time out. These were: GALEN, the Foundational Model of Anatomy (FMA) and NCBI Organismal Classification. These ontologies were discarded and not processed any further.

Ontologies With Non-Trivial Entailments There were 72 BioPortal ontologies that contained at least one non-trivial entailment. The list of these ontologies may be found in [10] and is available online as a summary[7]. For these ontologies, the average number of logical axioms (i.e. non-annotation axioms) per ontology was 10,645 (SD=31,333, Min=13, Max=176,113). The average number of non-trivial entailments per ontology was 1,548 (SD=6,187, Min=1, Max=49,537). The expressivity of the BioPortal ontologies with non-trivial entailments ranged from $\mathcal{EL}$ and $\mathcal{EL}$++ (corresponding to the OWL2EL profile) through to $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$ (the full expressivity of OWL 2 DL).

In summary, the ontology corpus provided by the BioPortal exhibits varying numbers of non-trivial entailments with a wide range of expressivities. It reflects current modelling practices and the kinds of ontologies that people use in tools.

4 Method and Results

All of the experiments detailed below were carried out using Pellet version 2.2.0[8]. For ontology loading, manipulation and reasoner interaction, the OWL API [11] version 3.2.2 was used. The OWL API has support for manipulating ontologies

[7] `http://www.stanford.edu/~horridge/publications/2012/iswc/justextract/data/bioportal-corpus-non-trivial-entailment-summary.pdf`

[8] The primary reason for using Pellet was that Pellet provides robust implementations of the OWL API reasoner interfaces and has reliable support for setting timeouts—a feature that is crucial for long running experiments.

at the level of axioms, and so it is entirely suited for the implementation of the justification finding algorithms.

Having introduced the overall test setup, the experiment is now described in detail.

Algorithm Implementation The black box algorithm for finding all justifications for an entailment, and its sub-routine algorithms, presented in [10] (Algorithms 4.1, 4.2, 4.5 and 4.6) were implemented in Java against the OWL API version 3.2.0. In essence this algorithm (Algorithm 4.1) is the de-facto standard *black box* algorithm for computing all justifications for an entailment. It does this by constructing a hitting set tree, using standard hitting set tree optimisations such as node reuse, early path termination etc. Its sub-routine algorithm for finding single justifications (Algorithm 4.2) uses the expand contract technique, where expansion is incremental, done by modularisation and selection function (Algorithm 4.5), and contraction is done using a divide and conquer strategy (Algorithm 4.6).

Test Data The test data consisted of the 72 BioPortal ontologies that contained non-trivial entailments. For each ontology, the set of *all* non-trivial direct subconcept ($A \sqsubseteq B$) and direct concept assertion ($A(a)$) entailments were extracted and paired up with the ontology.

Method The experiments were performed on MacBook Pro with a 3.06 GHz Intel Core 2 Duo Processor. The Java Virtual Machine was allocated a maximum of 4 GB of RAM. Pellet 2.2.2 was used as a backing reasoner for performing entailment checks, with each entailment check consisting of a load, followed by a query to ask whether or not the entailment held. The algorithm implementation described above was used to compute all justifications for each non-trivial entailment for each ontology. For each entailment, the CPU time for computing all justifications was measured, along with the number and sizes of justifications. For the sake of practicalities, because some ontologies have tens of thousands of non-trivial entailments (e.g. 49,000+ entailments for the coriell-cell-line ontology), a soft time limit of 10 minutes was imposed on computing all justifications for any one entailment. Additionally, an entailment test time limit of 5 minutes was placed on entailment checking.

Results[9] Figure 1 shows a percentile plot for time to compute all justifications. Note that mean values for each ontology are shown as transparent bars with white outlines. The x-axis, which shows Ontology Id, is ordered by the value of the 99th percentile. This percentile was chosen because it provides a good picture of how the algorithm will perform in practice for the vast majority of entailments. It also draws out the remaining 1 percent of outliers rather clearly.

There were *seven* ontologies that contained one or more entailments for which it was *not* possible to compute *all* justifications. These ontologies, along

[9] Large scalable plots of figures, and a spreadsheet containing the data used to generate them can be found at `http://www.stanford.edu/~horridge/publications/2012/iswc/justextract/`.

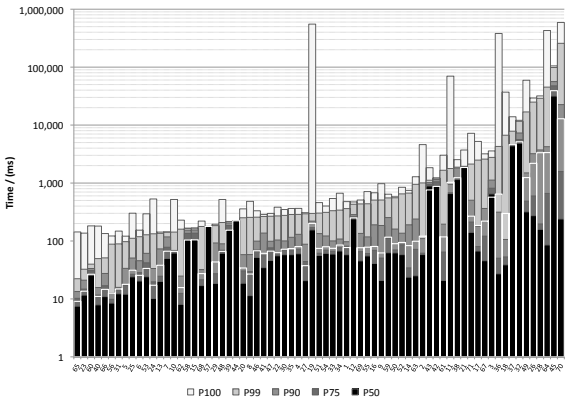

Fig. 1: Percentile and Mean Times to Compute All Justifications. Mean times are shown in white outlines. The x-axis (Ontology) is sorted by the 99th percentile (P99).

with the total number of entailments and the number of failed entailments are shown in Table 1. Table 1 also shows the mean/max number of justifications and mean/max justification size (number of axioms per justification) per failed entailment and the mean entailment checking times per failed entailment. In these seven ontologies there were three ontologies for which the failures occurred over less than one percent of entailments tested, a further three ontologies where the failures occurred for less than 7 percent of entailments tested, and one final ontology, where failures occurred for almost 75 percent of entailments tested. In this last ontology *all* of the failures were due to *entailment checking timeouts*. The failures relating to all of the other ontologies were due to timeouts during *construction of the hitting set tree*, which became too large to search within a period of 10 minutes.

Figure 2 and Figure 3 provide a picture of the justification landscape for the BioPortal ontologies. Figure 2 shows the the mean number of justifications per entailment per percentile. It should be noted that the percentiles are calculated from a *reverse* ordering of entailments based on justification size. That is, the nth percentile contains n percent of entailments that have the *largest* number of justifications. The x-axis in Figure 2 is ordered by the mean value of the 100th

Table 1: Black-Box Find All Timeouts

Ont.	Ents.	Failed	% Failed	Computed Justifications				Entailment Check Time / (ms)		
				Number		Size				
				Mean	Max	Mean	Max	Mean	SD	Max
19	49537	16	0.03	40.6	65	15.7	23	1.1	0.4	3
36	2230	1	0.04	414.0	414	13.5	17	0.5	0.3	2
64	566	4	0.71	1284.5	1411	25.1	28	2.5	1.6	13
70	3997	188	4.70	448.5	1060	12.7	24	7.0	77.8	141,721
45	148	9	6.08	1.6	2	21.9	24	1,979.9	6,762.4	41,840
3	44	3	6.82	1271.3	1494	26.9	35	2.5	1.3	10
32	35	26	74.29	2.2	7	2.5	8	2,601.2	23,781.0	270,004

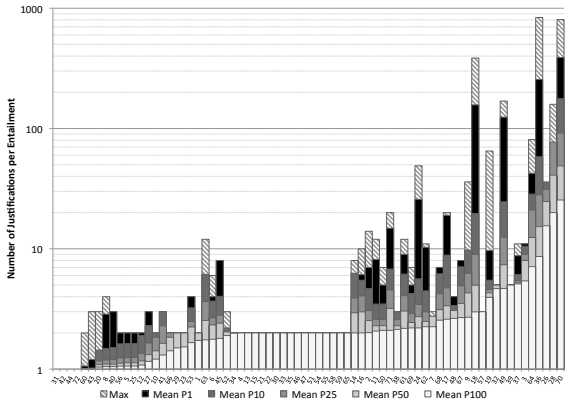

Fig. 2: The Mean Number of Justifications per Entailment for Various Percentiles. Percentiles of the Entailments Sorted by the Number of Justifications in Descending Order—e.g. P10 represents the top 10 percent of entailments with the highest number of justifications per entailment.

percentile (i.e. mean number of justifications per entailment). Figure 3 shows the mean number of axioms per justification per percentile along with the maximum number of axioms per entailment. The percentiles are calculated from a reverse ordering on justification size. For example, the nth percentile contains n percent of justifications that have a mean size *greater* than the mean of that percentile.

5 Analysis and Discusssion

The Practicalities of Computing All Justifications for An Entailment Out of the 72 ontologies it was possible to compute all justifications for all direct atomic subconcept and concept assertion entailments in 65 ontologies. There were seven ontologies that contained some entailments for which not all justifications could be computed. These failures are discussed below, however, the results from this experiment provide strong empirical evidence that it is largely practical to compute all justifications for these kinds of entailments in the BioPortal ontologies. Although the results cannot be statistically generalised to ontologies outside of the BioPortal corpus it is reasonable to assume that the results are suggestive for other real world ontologies.

Reasons for Failures Seven of the 72 ontologies contained entailments for which not all justifications could be computed. Broadly speaking there were two reasons for this: (1) The justifications for each failed entailment were numerous and large in size. This resulted in the size of the hitting set tree growing to a limit where it was not possible to close all branches within 10 minutes. In particular, for Ontology 36 the hitting set tree grew to over 3 million nodes, and for Ontology 70 the hitting set tree grew to over 1.6 million nodes. This compares to hitting set tree sizes in the tens of thousands for successful entailments. (2) Entailment checking performance was such that the number of entailment

checks, in combination with the time for each check, made it impossible to construct the hitting set tree within 10 minutes. This was the case with Ontology 45 and Ontology 32, both of which had average entailment checking times that were three orders of magnitude higher than for other ontologies. This problem was particularly endemic for Ontology 32, which suffered the largest number of failures, and had the worst entailment checking performance of all ontologies (M=2,601.2 ms, SD=23,781.0 ms, MAX=270,004 ms). Leaving aside entailment checking performance problems, which can be regarded as being out of the scope of control of this work, the number of justifications that were computable for failed entailments was very high. For example, for Ontology 3, which percentage-wise suffered the highest number of failures, the implementation was still able to compute on average 1271 justifications per failed entailment, with a maximum of 1494 justifications. With the exception of Ontology 32, which had a very high percentage of failures due to poor entailment checking performance, it is fair to say that over the whole corpus, and within individual ontologies, the failure rate is low to very low, thus indicating the robustness of the algorithms on real world ontologies. When the algorithm does fail to find *all* justifications, it is still possible to find *some* justifications, and the number of found justifications tends to be very large.

The Acceptability of Times for Computing All Justifications As can be seen from Figure 1, the majority of ontologies contained entailments for which all justifications could be computed within 1 second. For all but six ontologies, it was possible to compute all justifications for 99 percent of entailments within 10 seconds. Only two ontologies required longer than one minute for computing all justifications for 99 percent of entailments in these ontologies, with 90 percent of entailments in these ontologies falling below the one minute mark. It is clear to see that there are some outlying entailments in the corpus. In particular, Ontology 19 (the Coriell Cell Line Ontology) contains the most significant outlier, with one percent of entailments in this ontology requiring almost 150 seconds for computing all justifications. However, it appears that the times are perfectly acceptable for the purposes of generating justifications for debugging or repair in ontology development environments.

The Number of Justifications per Entailment As can be seen from Figure 2, the number of justifications per entailment varied over much of the BioPortal corpus. There were just four ontologies which had on average one justification per entailment. Even ontologies with low average numbers of justification per entailment did exhibit some entailments with large numbers of justifications as evidenced by the band of ontologies from 60 to 52 on the left hand side of Figure 2. On the right hand side of Figure 2, the band of ontologies from 14 through to 70 represent ontologies with very large numbers of justifications per entailment. For example, Ontology 70 had on average 25 justifications per entailment, with 10 percent of entailments having over 177 justifications, and 50 percent of entailments having over 48 justifications. This was closely followed by Ontology 28, which had on average 20 justifications per entailment, with 50 percent of entailments having over 40 justifications. The maximum number of

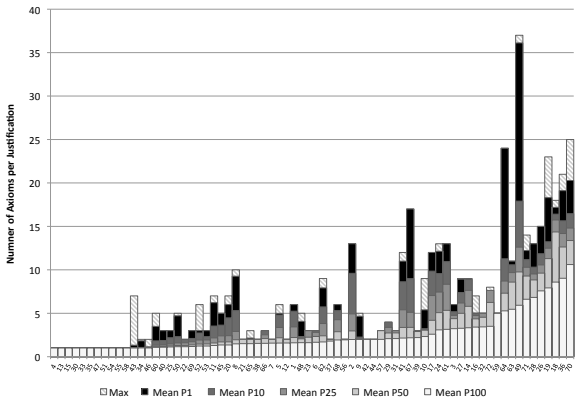

Fig. 3: Mean Number of Axioms per Justification. Percentiles of the Entailments Sorted by the Number of Justifications in Descending Order—e.g. P10 represents the top 10 percent of entailments with the highest number of justifications per entailment.

justifications for any one entailment occurred in Ontology 36, which had 837 justifications for one entailment. At this point it is worth noting that *none* of the empirical work detailed in the literature has uncovered ontologies with these large and very large numbers of justifications per entailment.

The Size of Justifications Figure 3 shows the mean numbers of axioms per justification per ontology. The ontologies are ordered by mean number of axioms per justification. There is a clear band of ontologies to the left hand side of Figure 3 that only have, on average, one axiom per justification. Recall that each entailment is a non-trivial entailment, which means that these justifications are not simply "self" justifications. In general the mean values (100th percentile) for each ontology are fairly low, with only 11 ontologies (shown on the right hand side of Figure 3) having over 5 axioms per justification on average . However, as witnessed by the 1st, 10th, 25th and 50th percentile columns in Figure 3, there are in fact many ontologies with many entailments that have larger numbers of axioms per justification. For example there are 10 ontologies where 50 percent of justifications contained over 7 axioms, and 10 percent of justifications contained 10 to 16 axioms. At the top end of the scale, several ontologies contained justifications with very large numbers of axioms. For example Ontologies 48, 50, 63, 18 and 41 contained justifications with 21, 23, 24, 25 and 37 axioms respectively. Finally it should be noted that these larger justifications do not simply consist of long chains of atomic subclass axioms. All in all, the number of justifications per entailment, and the size of justifications points to considerable logical richness being present in many ontologies in the BioPortal corpus.

6 Conclusions and Suggestions for Future Work

The detailed empirical investigation that has been carried out and presented in this paper provides strong evidence to conclude that computing all justifications

for direct class subsumption and direct class assertion entailments in the BioPortal corpus of consistent ontologies is practical. That is, for the vast majority of entailments in the majority of ontologies all justifications can be computed in under 10 minutes of CPU time. In essence, the justification finding algorithms used in the empirical evaluation here show good and robust runtime performance on realistic inputs.

While model based diagnosis techniques for computing all justifications fare extremely well, there are examples of entailments in realistic consistent ontologies for which it is not possible to compute all justifications. In these cases an incomplete solution, which could still consist of hundreds of justifications, must be accepted.

The number of justifications for entailments in naturally occurring domain ontologies can be very high. In the work presented here the number peaked at around 1000 justifications per entailment. The sizes of justifications in these ontologies can be very large, peaking at around 40 axioms per justification. The majority of ontologies with non-trivial entailments have multiple justifications per entailment with multiple axioms per justification.

In terms of future work, there are several strands that should be pursued. The first is that the experiments described here should be replicated and verified by third parties. It is reasonable to assume that the results presented here will be repeatable with other OWL reasoners, but this should ideally be investigated and verified. Finally, the experiments should be carried out on different ontology corpora. At the time of writing, third party non-biomedical-ontology installations of the BioPortal software are coming online. It would be interesting to compare the repositories of ontologies from different communities, in terms of non-trivial entailments, number justifications per entailment, size of justifications etc. and see how the justificatory structure and modelling style varies from one community to another.

References

1. Readings in Model Based Diagnosis. Morgan Kaufmann Publishers Inc. (1992)
2. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. Journal of Logic Computation 20(1), 5–34 (2010)
3. Cuenca Grau, B., Halasheck-Wiener, C., Kazakov, Y.: History matters: Incremental ontology reasoning using modules. In: ISWC 2007 + ASWC 2007
4. Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Ma, L.: Scalable semantic retrieval through summarization and refinement. In: AAAI 2007
5. Glimm, B., Rudolph, S., Völker, J.: Integrated metamodeling and diagnosis in OWL 2. In: ISWC 2010
6. Golbreich, C., Horrocks, I.: The OBO to OWL mapping, GO to OWL 1.1! In: OWLED 2007
7. Grimm, S., Wissmann, J.: Elimination of redundancy in ontologies. In: ESWC 2011
8. Grove, M.: OWLSight. http://pellet.owldl.com/ontology-browser (October 2009)
9. Halaschek-Wiener, C., Katz, Y., Parsia, B.: Belief base revision for expressive description logics. In: OWLED 2006

10. Horridge, M.: Justification Based Explanation in Ontologies. Ph.D. thesis, School of Computer Science, The University of Manchester (2011)
11. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web 2(1), 11–21 (February 2011)
12. Horridge, M., Parsia, B., Sattler, U.: Explanation of OWL entailments in Protégé-4. In: Poster and Demo Track, ISWC 2008
13. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: ISWC 2008
14. Ji, Q., Haase, P., Qu, G., Hitzler, P., Stadtmoeller, S.: RaDON – repair and diagnosis in ontology networks. In: ESWC 2009
15. Ji, Q., Qi, G., Haase, P.: A relevance-directed algorithm for finding justifications of dl entailments. In: ASWC 2009
16. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. Ph.D. thesis, The Graduate School of the University of Maryland (2006)
17. Kalyanpur, A., Parsia, B., Hendler, J.: A tool for working with web ontologies. In: International Journal on Semantic Web and Information Systems. vol. 1 (2005)
18. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: ISWC 2007 + ASWC 2007
19. Knublauch, H.: Composing the semantic web: Explaining inferences. http://composing-the-semantic-web.blogspot.com/2007/08/explanining-inferences.html
20. Meyer, T., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In: AAAI 2006
21. Mungall, C.: OBO Flat File Format 1.4 syntax and semantics. ftp://ftp.geneontology.org/pub/go/www/obo-syntax.html (February 2011)
22. Noy, N.F.: BioPortal: Ontologies and integrated data resources at the click of a mouse. Nucleic Acids Research 37 (May 2009)
23. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence 32, 57–95 (1987)
24. Scharrenbach, T., d'Amato, C., Fanizzi, N., Grütter, R., Waldvogel, B., Bernstein, A.: Default Logics for Plausible Reasoning with Controversial Axioms. In: Bobillo, F. (ed.) URSW 2010 (2010)
25. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: IJCAI 2003
26. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging incoherent terminologies. Journal of Automated Reasoning 39, 317 – 349 (2007)
27. Shchekotykhin, K., Friedrich, G., Jannach, D.: On computing minimal conflicts for ontology debugging. In: ECAI 2008 (2008)
28. Smith, B.: The OBO Foundary: Coordinated evolution of ontology to support biomedical data integration. Nature Biotechnology
29. Stuckenschmidt, H.: Debugging OWL ontologies - a reality check. In: EON-SWSC 2008 (2008)
30. Suntisrivaraporn, B.: Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies. Ph.D. thesis, Technical University of Dresden (2009)
31. Suntisrivaraporn, B., Qi, G., Ji, Q., Haase, P.: A modularization-based approach to finding all justifications for owl dl entailments. In: ASWC'08 (2008)