Towards Understanding Reasoning Complexity in Practice

Francisco Martín-Recuerda Dirk Walther

Universidad Politécnica de Madrid, Spain
fmartinrecuerda@fi.upm.es
dirk.walther@upm.es

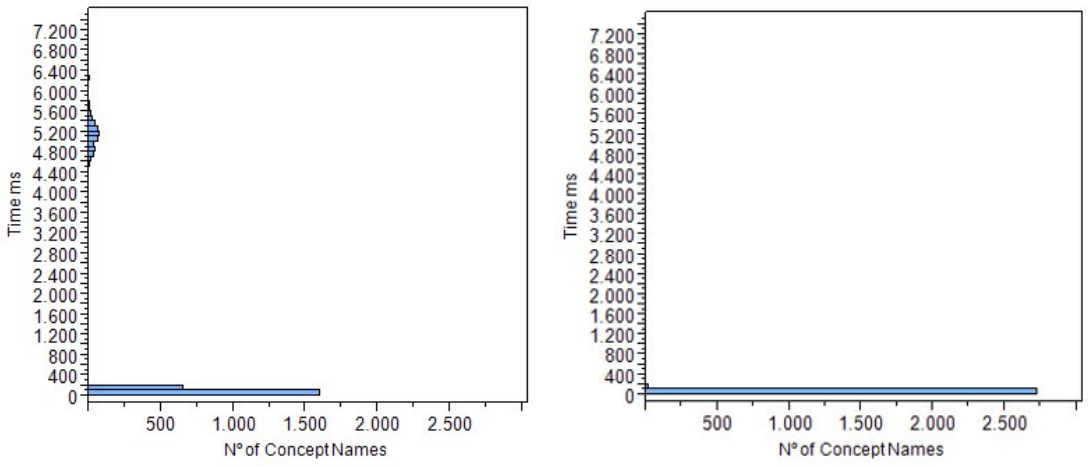
Abstract

Although the computational complexity of the logic underlying the standard OWL 2 for the Web Ontology Language (OWL) appears discouraging for real applications, several contributions have shown that reasoning with OWL ontologies is feasible in practice. It turns out that reasoning in practice is often far less complex than is suggested by the established theoretical complexity bound, which reflects the worst-case scenario. State-of-the reasoners like FACT++, HERMIT, PELLET and RACER have demonstrated that, even with fairly expressive fragments of OWL 2, acceptable performances can be achieved. However, it is still not well understood why reasoning is feasible in practice and it is rather unclear how to study this problem. In this paper, we suggest first steps that in our opinion could lead to a better understanding of practical complexity. We also provide and discuss some initial empirical results with HERMIT on prominent ontologies.

1 Introduction

We start our exposition with an experiment that shall illustrate and motivate our cause. The experiment shows that the size of an OWL ontology and the expressivity of the language in which it is formulated does not fully determine the time a reasoner takes to compute the answer to a reasoning problem. As input ontologies we use two variants of the well-known biomedical ontology GALEN: GALEN-UNDOCTORED and its modified version GALEN-DOCTORED [19]. In the experiment, we measure for each concept name of the ontology, the time needed to determine whether or not it is satisfiable wrt. the ontology. We choose the state-of-the-art reasoner HERMIT[1] to perform this task. We observe that for the majority of concept names the reasoner is very fast, but, surprisingly, also that it takes significantly longer in some cases. The distribution of the reasoning times for GALEN-UNDOCTORED are shown in Figure 1(a) (the x-axis shows the time in milliseconds and the y-axis the number of concept names). HERMIT takes in 82% of the cases less than 150 ms, whereas it takes more than 5 sec. for the other 18%. In practice, depending on the requirements of the application, such a variance in reasoning time may be tolerated as long as hard cases occur relatively seldom. On the other hand, when confronted with strict real-time constraints, we may find it unacceptable. The situation is different for GALEN-DOCTORED, which is GALEN-UNDOCTORED but with 253 axioms removed (out of 24 091 axioms in total). The modification affected only the internal structure of the ontology but not its signature (i.e. the terms defined in it) nor the language it is represented in (i.e. all language operators are still used). As Figure 1(b) shows, all difficult cases have disappeared. All those concept names for which HERMIT took several seconds before to determine their satisfiability, they are now checked as quickly as all the other terms. That is, relatively small changes in the input can have a seemingly disproportionally large impact on reasoning performance. We reduced the size of the ontology by less than 1% which caused an over 50-fold speed-up of HERMIT for some terms (e.g., for the concept name MonsPubis). The observed phenomenon is not specific to HERMIT (and its particular optimisations), a similar change in reasoning time can be observed with other reasoners as well. Thus it seems that implementation details of reasoners cannot sufficiently explain the phenomenon, even though they can affect reasoning performance

[1] www.hermit-reasoner.com



(a) GALEN-UNDOCTORED (b) GALEN-DOCTORED

Figure 1: Distribution of reasoning times of HERMIT on variants of GALEN

in unexpected ways [14]. In this paper, we do not consider implementation details nor the actual speed of computation (by the hardware), we are mainly interested in the input of a reasoner and how it makes reasoning hard. We hypothesise that there is a notion that solely depends on the internal structure of an ontology and that correlates well with reasoning performance. With the help of this notion we may possibly be able to predict reasoner performance on an ontology.

Expressive logics facilitate the development of ontologies, but whether we find their computational complexity for reasoning acceptable depends on the application. The W3C currently recommends OWL 2 as standard for the Web Ontology Language (OWL). The standard is based on the Description Logic $\mathcal{SROIQ}$ (with data types), which was designed with the intent to overcome the expressive limitations of the logic $\mathcal{SHOIN}$ underlying the previous standard OWL-DL. The goal was to provide the ontology developer with any desirable (but reasonable) expressive means for easy and intuitive modelling as to make the $\mathcal{SROIQ}$ even more useful in practice [10]. In short, $\mathcal{SROIQ}$ extends the basic Description Logic $\mathcal{ALC}$ [1] with nominals and qualified number restrictions, and it allows to impose many powerful conditions on roles such as (ir)reflexivity, symmetry, transitivity and universality, and between roles like complex role inclusions and disjointness. It seems that in terms of expressivity $\mathcal{SROIQ}$ indeed does not leave much to be desired. But what can be said about its usefulness regarding reasoning? According to the well-known trade-off between expressivity and computational complexity, more expressivity tends to come at a higher expense of complexity. $\mathcal{SROIQ}$ is intractable in general, i.e., reasoning can be too time-consuming to be useful. According to the current consensus, however, at least for some applications, the benefit of $\mathcal{SROIQ}$'s expressive power outweighs its intractability. This is, even though its computational complexity was established to be a staggering N2EXPTIME-complete [11]. In the light of this result, complete reasoning with OWL 2 ontologies seems rather unpromising. One needs to keep in mind though, that such a complexity bound holds for all possible inputs, including the input most difficult to reason with. Fortunately we do not often deal with worst cases in practice, because we do not make full use of the expressive power available in the language. For instance, one sure way to reduce complexity of reasoning is to abstain from using certain language constructs, i.e. to work within a fragment of $\mathcal{SROIQ}$. But developers tend to use the full language when, otherwise, it would be detrimental to their modelling capability. Besides, we do not necessarily have to restrict ourselves to fragments in order to enjoy feasible reasoning. The expressivity of a logic does not fully determine the difficulty inherent in a particular ontology. This will depend too on what is actually formulated in the ontology.

This observation is well-known in the automated theorem proving community. Theorem provers solve problem instances formulated in First-Order Logic whose satisfiability problem is undecidable. No reasoner could possibly compute an answer on every input. Even though there is no hope for complete reasoning, sound reasoning is feasible for many problem instances as is demonstrated with the library with Thousands of Problems for Theorem Provers (TPTP)[2]. Every problem is rated according to its difficulty and this rating is determined according to the actual performance of state-of-the-art reasoners [17]. Here the notion of

[2] http://www.tptp.org

difficulty depends on both the input and the reasoner. It remains unclear, however, what exactly makes a problem difficult.

The example above illustrates a variance in reasoning time when checking the satisfiability of terms in (variants of) GALEN using HERMIT. We argue that there is a more general principle behind this phenomenon. The performance of reasoners seems to largely depends on the internal structure of an ontology. We dub 'practical complexity' the complexity of reasoning with ontologies occurring in practice, where the ontologies contain a particular class of structures. An understanding of practical complexity seems to be relevant to developers and users of ontologies but also to developers of reasoners. It would have an impact on ontology design, maintenance and reuse. Understanding what makes reasoning hard can guide the modelling process of ontologies as to avoid hard reasoning. In this sense, efficiency becomes a modelling principle in ontology design. Existing ontologies may be optimised towards efficient reasoning by detecting and avoiding inefficient modelling. Ontologies may be designed to yield "efficient" modules, which may make them preferable candidates for reuse. By understanding efficient reasoning, users may choose a selection of ontologies to be imported that together yield efficient reasoning, and developers of reasoners may be guided towards implementing more efficient optimisations.

Much existing work in knowledge representation and reasoning seems relevant in this context. We benefit from a wealth of complexity theoretic results of knowledge representation formalisms, primarily Description Logics, that inspired the implementation of actual reasoning systems. A recently emerging area that received much interest is the investigation of the modular structure of ontologies to facilitate their reuse [7] and also approaches relevant for ontology versioning that study the logical difference and incremental reasoning [13, 4]. The notion of modularity plays a key role in these approaches [7, 12]. However, these lines of research do not consider, at least not explicitly, reasoning complexity in practice and provide explanations for it.

Empirical results have shown that pathological problems that would lead to worst-case behaviour rarely occur in realistic applications [9]. Several efforts have been made to produce experimental settings [5, 14, 2] that facilitate automated evaluation of OWL reasoners. However, performance is measured following a black-box approach, where the implementation details of the reasoner are abstracted away and only external variables are considered. These settings are used to identify and classify hard cases, but they say little about why they are hard. To improve this situation, applications have been created that monitor the inner workings of a reasoner during execution of a reasoning task, e.g. 'Tweezers' [18] for PELLET or a Protégé plug-in for model exploration based on FACT++ [3]. These implementations seem useful for studying reasoning complexity in practice, but we are not aware of any systematic study based on them. An alternative approach is to use justifications, which are minimal subsets of an ontology implying that a certain concept is satisfiable [8, 2]. The approach is interesting, however, so far it is not clear how reasoning complexity can be explained.

In this paper, we suggest a different, empirical approach for a systematic study of reasoning complexity in practice. The general aim is to explain what makes reasoning hard. The actual time needed to perform the reasoning task depends (besides the hardware) on the reasoner and its input. We therefore investigate the correlation of reasoning time and other metrics internal to the reasoner (thus avoiding a black-box approach) with various syntactic features of the input ontology. Here we focus solely on the reasoning task that determines whether or not a concept name is satisfiable wrt. an ontology. Regarding the syntactic features, a manual inspection of ontologies seems infeasible, especially with ontologies containing thousands of axioms. Instead we employ modularisation techniques to determine the relevant part of the ontology involved in a term's definition. After extracting a module for a term, we investigate the syntactic properties of the module and study in how far they correlate with the reasoning time. It turns out, for instance, that the size of the modules is not a sufficient criterion for telling apart "difficult" from "easy" terms, on which the reasoner is respectively slow or fast. Modules of "easy" terms can be as large as the modules of "difficult" terms, and *vice versa*.

The paper is organised as follows. We skip a preliminary section, where we would formally define the notion of ontologies and modules, the Description Logics they are represented in and the reasoning tasks; instead we refer to the relevant literature. In Section 2, we give more details about our empirical approach to studying practical complexity, i.e., how we measure the performance of HERMIT and how we analyse ontologies syntactically. We continue with discussing the results of the experiments with real-world and artificial ontologies in sections 3. The paper closes with conclusions in Section 4. Full details of the experiments and their evaluation can be found in a long version of the paper.

2 Description of Approach

In this section, we describe in more detail our empirical approach to studying reasoning complexity in practice. We start with describing what appears to be the main difficulties for such an approach. An analysis of reasoner behaviour is complicated as reasoners are often badly documented even though they have become sophisticated applications using optimisations and heuristics (triggered by certain language constructors). Source code may not be available under open source licenses. Tool support for collecting execution details of reasoners is rarely available or not well-documented. Configuration details in the test bed (e.g., the operating system chosen) or in the OWL reasoner (manual activation or deactivation of certain optimizations) can significantly affect the performance measured. With few exceptions like the Tones ontology repository[3], BioPortal[4] and the SEALS infrastructure[5] there are no services available for researchers for searching and retrieving OWL ontologies that have been previously curated and properly documented (annotated). Moreover, we rely on automated tools as ontologies are often too large for manual inspection.

We now fix some parameters to narrow down the scope for the study of reasoning complexity in practice. We start our investigation within the following setting. Instead of dealing with several OWL reasoners that implement different algorithms together with particular optimisations and heuristics, we focus on one reasoner only, namely HERMIT v1.3.4. In all experiments we use HERMIT to check for the satisfiability of concept names wrt. ontologies. HERMIT is a state-of-the-art OWL 2 reasoner able to process $\mathcal{SROIQ}$-ontologies [16]. The reasoning algorithm is a hybrid of resolution and tableau and it is optimised to reduce *OR-* and *AND-branching* during the creation of the tableau, which have been identified to be main sources of complexity. In fact, the algorithm exhibits no non-determinism on ontologies such as GALEN, NCI and SNOMED CT. These ontologies and GO are frequently used for evaluation purposes [19] and we use them not at least because of their relatively weak languages, that would trigger less optimisations and heuristics of HERMIT. We also benefit from HERMIT's open source license and its built-in its debugging and monitoring facilities. The lack of documentation was compensated by the readiness of the developers to discuss implementation details. Of course, these restrictions only partially simplify the problem as we still have to deal with large ontologies.

In the experiments, we will measure the time needed by HERMIT to perform a SAT-test of a concept name wrt. an ontology. Moreover, we employ the built-in profiler to collect other execution details to see how exactly HERMIT spends its time reasoning. On the other hand, we investigate how the concept name under consideration is defined and related to other terms in the ontology. To this end, we extract the so-called $\bot$-*local module* for the concept name [6] using the 'Locality Module Extractor'[6]. Tool support was decisive for the choice of module, even though there may be irrelevant axioms in these modules [7]. We study the syntactic characteristics of each module and investigate in how far they correlate with the execution details of the reasoner.

To characterise the reasoning complexity we commonly rely on time and memory consumption of the reasoner, but we will also consider other metrics. Note that we decided to exclude the time needed for HERMIT to load and preprocess the input ontology. This may not be acceptable when comparing the performance of several reasoners. To minimize noise in the measures, we repeated the SAT-test for a concept names 100 times and we calculated the median of the reasoning times collected for each test (cf. '*median SAT-time*' in Table 2). We do not use the mean (or average) of the reasoning times as this notion is more susceptible to extreme values, which we observe in our experimental setup; see, e.g., [2] for more discussion on mean vs. median. It is well-known that OR-branching (i.e. the exploration of non-deterministic choices) in a tableau, is fairly expensive. For studying this specific phenomenon with HERMIT, we monitor the number of non-deterministic choices explored ('*# branching points*') and the number of backtracks ('*# backtracks*'). For the measuring memory consumption, we only consider the number of nodes in the tableau produced by the reasoner ('*# tableau nodes*'). Apart from the reasoner, we consider a selection of syntactic properties of the $\bot$-local module that includes all the axioms needed for the definition of the concept name. For instance, we consider the expressivity of a module, the number and kind of axioms it contains, and the size and constituent parts of its signature; cf. Table 2. The goal is to analyse how these variables are correlated .

Next to the ontologies mentioned above, we create artificial ontologies of different sizes by instantiating certain axiom pattern. Then we study how the performance of HERMIT is affected by inputs of different

[3] http://owl.cs.manchester.ac.uk/repository/
[4] http://bioportal.bioontology.org
[5] http://www.seals-project.eu
[6] http://krono.act.uji.es/people/Ernesto/safety-

Name	Description	Axiom Scheme
axiom chain	$\sqsubseteq$–chain of concept names of length n whose final concept is subsumed by the bottom concept	$\mathsf{A}_i \sqsubseteq \mathsf{A}_{i+1}$ $\mathsf{A}_n \sqsubseteq \bot$
axiom cycle	$\sqsubseteq$–cycle of concept names of length n	$\mathsf{A}_i \sqsubseteq \mathsf{A}_{i+1}$ $\mathsf{A}_n \sqsubseteq \mathsf{A}_0$
role chain	role chain of length n whose leaf concept is subsumed by the bottom concept	$\mathsf{A}_i \sqsubseteq \exists r.\mathsf{A}_{i+1}$ $\mathsf{A}_n \sqsubseteq \bot$
role cycle	role cycle of length n	$\mathsf{A}_i \sqsubseteq \exists r.\mathsf{A}_{i+1}$ $\mathsf{A}_n \sqsubseteq \mathsf{A}_0$
binary tree	binary tree of depth n whose leaf concepts are all subsumed by the bottom concept	$\mathsf{A}_i \sqsubseteq \exists r.\mathsf{A}_{i+1} \sqcap \exists s.\mathsf{A}_{i+1}$ $\mathsf{A}_n \sqsubseteq \bot$
binary tree (cyclic)	binary tree of depth n whose leaf concepts are all subsumed by the root concept	$\mathsf{A}_i \sqsubseteq \exists r.\mathsf{A}_{i+1} \sqcap \exists s.\mathsf{A}_{i+1}$ $\mathsf{A}_n \sqsubseteq \mathsf{A}_0$

Table 1: Ontology pattern of size $n \geq 0$ (i ranges over $\{0, \ldots, n-1\}$)

sizes and compare this with the performance on real-world ontologies. Table 1 presents the definition of the patterns together with a description.

3 Evaluation of the Experiments

In this section, we discuss the results of the experiments. We first describe the experiments with the real ontologies and then the ones with the artificial ontology pattern.

3.1 Experiments with Real Ontologies

For each of the ontologies considered, Table 2 provides the values for the hardest satisfiability test executed and the maximum values that we have obtained for a selection of the metrics considered.

Metrics of HermiT	Ontologies									
	Galen-undoc.		Galen-doc.		GO		NCI		Snomed ct	
	hardest test	max value	hardest test	max value	hardest test	max value	hardest test	max value	hardest test	max value
median SAT-time (ms)	7 439	7 439	176	176	14	14	250	250	1 389	1 389
# branchings points	-1	-1	-1	-1	-1	-1	-1	6 152	-1	-1
# backtracks	0	0	0	0	0	0	0	0	0	0
# tableau nodes	17 317	17 317	1 368	1 540	17	149	137	403	2 055	2 055
Metrics of modules										
expressivity	$\mathcal{ALEHIF}+$		$\mathcal{ALEHIF}+$		$\mathcal{ALER}+$		$\mathcal{ALC}$	$\mathcal{ALCH}$	$\mathcal{ALEH}$	$\mathcal{ALEH}$
# axioms in	2 381	2 776	940	1 040	211 176	211 465	412	7 862	642	1 278
# $\equiv$-axioms	131	202	78	98	0	0	1	1 931	149	264
# $\sqsubseteq$-axioms	1 231	1 449	279	386	54	259	205	2 378	204	469
signature size	712	844	349	389	45	129	169	3 516	286	546
# concept names	551	655	226	256	34	118	158	3 463	275	534
# role names	161	195	123	133	2	4	11	53	11	74
# individual names	0	0	0	0	9	9	0	0	0	0

Table 2: Execution details of HermiT vs. syntactic features of $\bot$-local modules

For all ontologies (except NCI), the maximum values for the metrics '*# branching points*' and '*# back-*

tracks' are always -1 and 0, respectively. This means that HERMIT works deterministically on those ontologies, i.e., it did not have to handle disjunctions during reasoning (*OR-branching*). These ontologies are expressed in the Description Logic Horn-$\mathcal{SHIQ}$ and they can be translated into Horn-clauses without "unnecessary" non-determinism [16]. OR-branching is usually very expensive in terms of computational resources and many optimizations for OWL reasoners have been designed to minimize it [9].

Regarding OR-branching, the results for NCI are also interesting. The maximum values for the metrics '*# branching points*' is '6 152', for '*# backtracks*' is 0, and for '*# tableau nodes*' (used for building the tableau) is 403 (cf. the right part of the column for NCI in Table 2). So, HERMIT has explored thousands of disjunctions without performing any backtrack and keeping the size of the tableau relatively small. According to the developers of HERMIT, this is due to an optimisation called *individual reuse* that implements existential expansion non-deterministically with the goal of creating smaller models and significantly reducing backtracking [15].

Table 2 also discovers an interesting behaviour of HERMIT for the Gene Ontology (GO). The hardest concept satisfiability test requires 14 ms and the '*# tableau nodes*' for building the tableau are 17. However, the maximum value for '*# tableau nodes*' is 149. Looking at the results of other concept satisfiability tests, we found similar cases that required less than 14 ms, but allocated more than 17 nodes for building the tableau. Two possible explanations where suggested by the developers of HERMIT. The first one is that the results were affected by the *noise* of the test environment (i.e., insufficient precision when measuring time, background processes, etc.). As checking satisfiability of concept names in GO take relatively little time, the time measured is easily distorted by external factors. However, re-running the experiments several times yielded similar results. A possible explanation is that HERMIT has to deal with millions of axioms in GO that slows down the performance due to searching for applicable clauses. This issue requires further investigation.

Another aspect worthy to discuss is the perception of the hardness of an ontology. Comparing the maximum values of '*median SAT-time*' for each of the ontologies considered in this study, GALEN-UNDOCTORED is the ontology with the highest value, 7 439 ms. This value is about seven times higher than the highest value for SNOMED CT, 1 389 ms. But SNOMED CT is about 146 times larger than GALEN-UNDOCTORED and the language used for defining SNOMED CT is slightly less expressive. We see that difficulty of an ontology is not determined by the language an ontology is formulated in or its size, but how terms are defined, i.e. the internal structure of the ontology.

As it was mentioned in the introduction, it is also interesting to see the impact on the performances of HERMIT when 243 axioms were deleted in GALEN-UNDOCTORED to create GALEN-DOCTORED. In the case of the former, the hardest concept satisfiability test requires 7 439 ms and 17 317 nodes where allocated for building the tableau. For GALEN-DOCTORED, however, the hardest concept satisfiability test requires 176 ms (42 times less) and only 1 368 nodes in the tableau were allocated (13 times less). That is, the reasoner now creates a much smaller tableau and requires much less time for it. This is because removing the axioms causes HERMIT to apply fewer existential expansions. For a more precise explanation, we need to study the internal behaviour of HERMIT more closely.

Finally, we investigate in how far the metrics correlate to one another. A strong correlation between metrics can indicate a causal relation between them possibly explaining what makes reasoning hard. Table 2 contains only fairly simple syntactic metrics that are related with the size of modules. Unfortunately, we did not find strong correlations between the input metrics and the metrics of the reasoner. However, this negative result is preliminary and we cannot conclude that there are no correlations between the certain syntactic properties and reasoning time. We plan to extend the syntactic metrics to additionally take into account the internal structure of the ontologies.

Next we evaluate the experiments with ontologies whose structure is well-understood – the artificial ontology pattern.

3.2 Artificial Ontology Patterns

Here we evaluate the experiments with the six ontology patterns defined in Table 1. We are interested in the degree of difficulty of a pattern, i.e., in how far reasoning time increases with increasing size of pattern instances. Table 3 shows the time HERMIT needs to determine whether the concept name A_0 is satisfiable wrt. instances of the patterns. The size of the pattern instances ranges over 1 000, 5 000, 10 000, 50 000 and 100 000. Note that we restrict ourselves to pattern size here, but, given the pattern size and pattern

definition in Table 1, we can easily determine the actual size (i.e. counting all symbols) of the respective pattern instance.

Pattern size n	SAT-test time (in ms)					
	axiom chain	axiom cycle	role chain	role cycle	binary tree	binary tree (cyclic)
1 000	1	5	8	8	23	23
5 000	5	5	172	170	389	403
10 000	11	11	632	634	1 646	1 647
50 000	121	122	32 939	33 001	109 983	82 976
100 000	205	225	152 352	151 253	344 846	345 381

Table 3: SAT-times of HERMIT on ontology patterns of increasing size

We observe that both pattern '*axiom chain*' and '*axiom cycle*' are far easier to reason with than all the other patterns. The reasoning time grows nearly linearly with the input size and, as expected, the size of the tableau is always 1. The other patterns cause a significantly stronger increase in reasoning time with increasing pattern size. The time for the patterns '*binary tree*' and '*binary tree (cyclic)*' grows 2-3 times faster than '*role chain*' and '*role cycle*'. We notice that cyclicity (as modelled in the patterns) does not much affect reasoning time, but we suspect the depth of role nesting to have a significant impact.

4 Conclusion

We are interested in explaining reasoning complexity in practice, where we often do not encounter computational worst-case scenarios. The goal of this paper is to define and validate the initial stages of a more broad study that contribute to improve our understanding of reasoning complexity in practice. To this end, we introduced several restrictions to simplify the problem in order to facilitate a systematic empirical study. We investigated practical complexity by performing some experiments with prominent biomedical ontologies such as GALEN, GO, NCI and SNOMED CT using the OWL 2 reasoner HERMIT. As reasoning task we focussed on the satisfiability test of a concept name wrt. an ontology. Clearly, the actual time needed to perform the reasoning task depends (besides the hardware) on the reasoner and the input. We had a closer look at both, i.e., we studied various variables ranging over syntactical features of the input ontologies (the module for the concept name to be precise) and over some performance details from HERMIT (obtained via its internal profiling function). The goal is to check in how far any of these variables correlate, possibly indicating a causal relation between them, which may help us to identify what makes reasoning hard. It turns out that reasoning time cannot sufficiently explained with the syntactic features of modules that were considered in this work. This finding suggests to consider more refined structural information of the ontologies. We leave this for future work as well as contrasting these findings with the performance of other reasoners, e.g., PELLET. We also investigated the time required by HERMIT to reason with various simple ontology pattern. The experiments show that module size for a concept name is not sufficient to predict a certain time needed to check for satisfiability. The internal structure of the ontology needs to be taken into account as well.

We also conclude that the built-in profiler of HERMIT can be used as an analytical tool for a better understanding of practical complexity. Unfortunately, due to the lack of documentation, this tool should be hardly accessible for non-developers.

Acknowledgements

This work has been supported by the EU project SEALS (FP7-ICT-238975), the 'Juan de la Cierva' program (MICINN-JDC, ref. JCI-2009-05801) and by the SAS Institution[7]. We would also like to thank Birte Glimm, Boris Motik and Ernesto Jiménez-Ruiz for their valuable support.

[7] http://www.sas.com

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] S. Bail, B. Parsia, and U. Sattler. Justbench: a framework for owl benchmarking. In *Proceedings of ISWC'10*, pages 32–47, Springer, 2010.

[3] J. Bauer, U. Sattler, and B. Parsia. Explaining by example: Model exploration for ontology comprehension. In *Proceedings of DL'09*. CEUR-WS, 2009.

[4] B. Cuenca Grau, C. Halaschek-Wiener, Y. Kazakov, and B. Suntisrivaraporn. Incremental classification of description logics ontologies. *Journal of Automated Reasoning*, 44:337–369, 2010.

[5] T. Gardiner, D. Tsarkov, and I. Horrocks. Framework for an automated comparison of description logic reasoners. In *Proceedings of ISWC'06*, pages 654–667. Springer, 2006.

[6] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: extracting modules from ontologies. In *Proceedings of WWW'07*, pages 717–726. ACM, 2007.

[7] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: theory and practice. *Journal of Artificial Intelligence Research*, 31:273–318, February 2008.

[8] M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in owl. In *Proceedings of ISWC'08*, pages 323–338. Springer, 2008.

[9] I. Horrocks. Implementation and optimisation techniques. In *The Description Logic Handbook: Theory, Implementation, and Applications*, Chapter 9, pages 306–346. Cambridge University Press, 2003.

[10] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In *Proceedings of KR06*, pages 57–67. AAAI Press, 2006.

[11] Y. Kazakov. RIQ and SROIQ are harder than SHOIQ. In *Proceedings of KR08*, pages 274–284. AAAI Press, 2008.

[12] B. Konev, C. Lutz, D. Walther, and F. Wolter. Formal properties of modularisation. In *Modular Ontologies*, LNCS, pages 25–66. Springer, 2009.

[13] B. Konev, D. Walther, and F. Wolter. The logical difference problem for description logic terminologies. In *Proceedings of IJCAR'08*, pages 259–274. Springer, 2008.

[14] M. Luther, T. Liebig, S. Böhm, and O. Noppens. Who the heck is the father of bob? In *Proceedings of ESWC'09*, pages 66–80. Springer, 2009.

[15] B. Motik and I. Horrocks. Individual Reuse in Description Logic Reasoning. In *Proceedings of IJCAR'08*, volume 5195 of *LNAI*, pages 242–258. Springer, 2008.

[16] B. Motik, R. Shearer, and I. Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009.

[17] G. Sutcliffe and C. Suttner. Evaluating General Purpose Automated Theorem Proving Systems. *Artificial Intelligence*, 131(1–2):39–54, 2001.

[18] T. D. Wang and B. Parsia. Ontology performance profiling and model examination: first steps. In *Proceedings of ISWC'07/ASWC'07*, pages 595–608. Springer, 2007.

[19] M. Yatskevich, G. Stoilos, I. Horrocks, and F. Martin-Recuerda. D11.1 Evaluation design and collection of test data for advanced reasoning systems. Technical report, 2009. `http://about.seals-project.eu/downloads/category/1-`