Benchmarking OWL Reasoners

Jürgen Bock[1], Peter Haase[2], Qiu Ji[2], Raphael Volz[1]

[1]FZI Research Center for Information Technologies, Karlsruhe, Germany
[2]Institute AIFB, University of Karlsruhe, Germany
{bock,volz}@fzi.de, {haase,qiji}@aifb.uni-karlsruhe.de

Abstract. The growing popularity of semantic applications makes scalability of ontology reasoning tasks increasingly important. In this work, we first analyze the ontology landscape on the web, and identify typical clusters of expressivity. Second, we benchmark current ontology reasoners, by using representative ontologies for each cluster and a comprehensive set of queries. We point out applicability of specific reasoners to certain expressivity clusters and reasoning tasks.

1 Introduction

Semantic applications based on ontologies have become increasingly important in recent years. Yet, scalability remains one of the major obstacles in leveraging the full power of using ontologies for practical applications. Reasoning with OWL ontologies has high worst complexity, but indeed many large scale applications normally only use fragments of OWL that are rather shallow in logical terms and do not require sophisticated reasoning algorithms. Surveying the landscape of existing ontologies, we observe a broad spectrum of ontologies that differ in terms of size, complexity and their ratio between terminological and factual assertions. Keet and Rodríguez [9] point out that there is a high demand for either very expressive ontology languages to represent rather complete knowledge, or less expressive ontology languages, which are more tractable w.r.t. reasoning or other computational tasks. This issue is often called the *computational cliff* and leads to the problem, that often only small fragments of ontology languages are exploited, where reasoners are used that are optimized to a larger number of features, and in particular those, that are actually not used in the fragment. Today, a number of different reasoners are available that are based on quite different design decisions in addressing the tradeoff between complexity and expressiveness on the one hand and scalability on the other hand: Classical description logic reasoners based on tableau algorithms are able to classify large, expressive ontologies as often found *e.g.* in the bio-medical domain, but they often provide limited support in dealing with large number of instances. Database-like reasoners that materialize inferred knowledge upfront are able to handle large amounts of assertional facts, but are in principle limited in terms of the logic they are able to support. Deciding for an appropriate reasoner for a given application task is far from trivial. In order to support such decisions, comparisons of reasoners based on benchmarks are required.

While a number performance evaluations for OWL reasoners have already been performed in the past, all of them so far targeted only special purpose tasks, *e.g.* focussing either on classical description logic reasoning tasks [4], or on answering conjunctive queries over large knowledge bases based on rather inexpressive ontologies [7]. In our work we aim to go a step further and intend to provide guidance for selecting the appropriate reasoner for a given application scenario. In order to do so, we provide a survey of the ontology landscape, discuss typical reasoning tasks and define a comprehensive benchmark. Based on the benchmark results we identify which reasoners are most adequate for which classes of ontologies and corresponding reasoning tasks.

The paper is organized as follows: In Section 2 we discuss related work on benchmarking OWL reasoners. Based on an overview of the ontology landscape and relevant language fragments provided in Section 3, we define our benchmark in Section 4. In Section 5, we give a description of the reasoners selected for our benchmark. In Section 6 we report on the experiments performed. We conclude with an outlook to future work in Section 7.

2 Related Work

With the availability of practical reasoners for OWL, a number of benchmarks for evaluating and comparing OWL reasoners have been proposed. The first one – the Lehigh University Benchmark (LUBM) – was proposed by Guo *et al.* [7]. LUBM concentrates on the reasoning task of answering conjunctive queries over an OWL Lite ontology with an ABox of varying size. It was later pointed out that – while the ontology itself is in OWL Lite – answering the proposed queries does not require OWL Lite reasoning, but instead can be performed by realizing the ABox, *i.e.* computing the most specific concept(s) that each individual is an instance of. This indeed is performed by many system benchmarks, including for example the evaluation of RacerPro [8]. In addition to measuring the performance of the reasoners, LUBM provides a measure for correctness of the reasoners, analyzing how many of the correct answers are returned (completeness) and how many of the returned answers are correct (soundness). However, we believe that such measures are not helpful in selecting a reasoner for a given task. Instead what is missing is a detailed analysis, which fragment of the OWL ontology language are actually needed for a given task and supported (correctly) by which reasoner.

[4] presents a system for comparing DL reasoners that allows users (a) to test and compare OWL reasoners using an extensible library of real-life ontologies; (b) to check the correctness of the reasoners by comparing the computed class hierarchy; (c) to compare the performance of the reasoners when performing this task. Again, this benchmarking system is only targeted to classical DL reasoning tasks, disregarding many other practical applications of OWL.

[11] provides a comparison of reasoning techniques with a focus on querying large DL ABoxes. The results show that, on knowledge bases with large ABoxes but simple TBoxes, the KAON2 algorithms for reducing a DL knowledge base to

a disjunctive datalog program show good performance; in contrast, on knowledge bases with large and complex TBoxes, existing techniques still perform better.

In [15] the authors pointed out some deficiencies with existing benchmarks and formulated requirements they would like to see met by new benchmarks. These requirements were driven by two major use cases: (1) Frequent ABox changes (situation classification) and (2) rare ABox changes (social networks). While in our work we rather concentrate on the dimensions of the classes of ontologies and reasoning tasks, we take most of the defined requirements into account (cf. Section 4.4).

We base our benchmark on an analysis of the ontology landscape, similar to the one conducted by Wang *et al.* [14], where our more recent analysis is based on the WATSON corpus[1] with a larger number of ontologies.

3 Overview of the Web Ontology Landscape

Ontologies on the web are becoming increasingly numerous and differ significantly in their expressivity, as well as in the size of their TBoxes and ABoxes. In order to identify a representative picture of the ontology landscape, we analyzed 3303 ontologies with particular respect to their expressivity. Ontologies were drawn from the WATSON corpus and expressivity determined using statistics provided by the SWOOP editor[2] and the KAON2 OWL tools[3]. We preferred SWOOP to determine expressivity, since it provides a more fine-grained breakdown of DL expressivity with respect to tractable fragments than *e.g.* Pellet does.

We made the observation, that out of the 6224 OWL ontologies recorded by WATSON only 3303 could have been loaded by both SWOOP and the KAON2 OWL tools. This mainly traces back to syntactical errors, a problem which has already been identified by d'Aquin *et al.* [3].

In this analysis, we identified four main expressivity fragments, namely the RDFS fragment of description logics[4], OWL DLP, OWL Lite, and OWL DL. Hereby, the fragments OWL DL and OWL Lite comprise ontologies that fall into description logics of at most $\mathcal{SHOIN}(D)$ for OWL DL, and $\mathcal{SHIF}(D)$ for OWL Lite, resp. In addition to these, we included the tractable fragment OWL DLP [6], since it retains a number of interesting features of OWL Lite while keeping the complexity low to a certain degree. Theoretical investigations proved the combined complexity for standard reasoning tasks in the DLP fragment to be EXPTIME, while the data complexity for both standard reasoning tasks and conjunctive query answering remains PTIME complete [5]. Figure 1 illustrates the inclusion of the respected fragments according to their expressivity.

The OWL Working Group[5] is currently also looking at other tractable fragments [5], which we do not address in this work. The reason is, that most of

[1] `http://watson.kmi.open.ac.uk/WatsonWUI/`
[2] `http://www.mindswap.org/2004/SWOOP/`
[3] `http://owltools.ontoware.org/`
[4] We will call this fragment RDFS(DL) subsequently.
[5] `http://www.w3.org/2007/OWL/wiki/OWL_Working_Group`

$$\begin{tikzcd}
 & \text{OWL DL} & \\
\text{OWL DLP} \arrow[ur] & & \text{OWL Lite} \arrow[ul] \\
 & \text{RDFS(DL)} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Fig. 1. Inclusion of language fragments of different expressivity.

these fragments are heavily discussed and not widely used up to now. Furthermore there are no mature reasoners available, that particularly deal with these fragments, whereas *e.g.* reasoners such as OWLIM can process DLP ontologies. For instance by investigating ontologies from WATSON that fall into the DL-Lite fragment, we noticed, that ontologies of this kind are mostly toy examples and hence not of significant importance for this benchmark. We also dropped the $\mathcal{EL}$ fragment, where SWOOP found only 3 ontologies.

The most lightweight complexity fragment we identified was RDFS(DL), which is, in terms of expressivity, sufficient for representing many taxonomy-style ontologies. Due to its simplicity it seems to be popular for many use cases of ontologies in current web based applications.

In DLP we go clearly beyond RDFS(DL) by allowing more specified properties, and a restricted form of intersection, universal and existential quantification. The DLP fragment was one of the smallest fragments we analyzed. This may be due to the more cautious ontology design that has to be followed, when it comes to the use of specific features that are not supported in DLP. The fact that DLP has only very recently been identified as a tractable fragment, also explains the smaller number of ontologies that have been found, since most DLP ontologies in our corpus have been identified to be only incidentally in the DLP fragment, but few might be explicitly designed as such.

The OWL Lite fragment is determined by a stronger axiomatization of classes, as the frequent use of restrictions in nearly all of those ontologies demonstrates.

In the OWL DL fragment we can find the full range of available OWL DL features, including nominals. This fragment contains the largest ontologies on average, however, OWL DL ontologies are still less frequent than OWL Lite ontologies on the web.

Figure 2 illustrates the ontology landscape according to the different fragments, in terms of number of ontologies and average number of classes in each fragment. As one can see, the RDFS(DL) fragment is the largest fragment, where ontologies make use of only the basic, taxonomic features, such as classes and basic properties. However, there is also a relatively small average number of classes in RDFS(DL) ontologies, which is mainly because of the large number of extremely small meta-data files present in the web as FOAF files, or RDF exports of semantic annotations of certain websites or wiki pages. SWOOP validates a large percentage (73.45 %) of RDFS(DL) ontologies are classified as OWL Full, which indicates a high syntactical error rate in these small meta-data files.

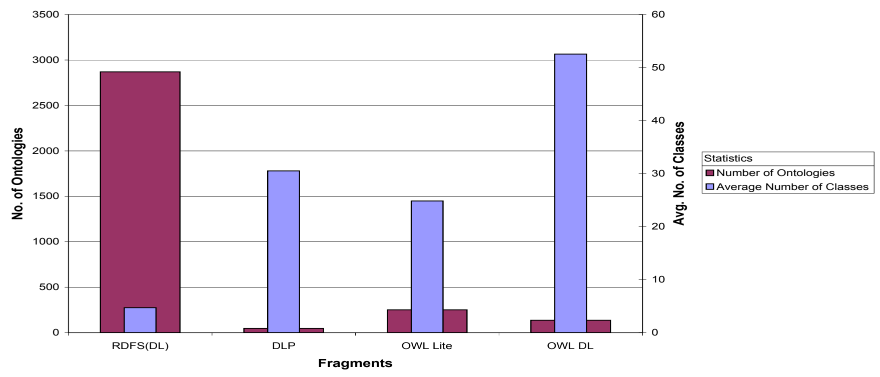

Fig. 2. Total number of ontologies and average number of classes by language fragments.

4 Benchmark Definition

4.1 Reasoning Tasks

Many reasoning tasks for OWL correspond to standard description logic reasoning tasks, *i.e.* tasks that allow to draw new conclusions about the knowledge base or check its consistency. Theoretically it is possible to reduce all reasoning tasks to the task of checking KB consistency. However in practice this is not necessarily the fastest way of reasoning and various optimizations are taken for different tasks. We therefore analyze reasoning tasks separately.

TBox reasoning tasks. Reasoning tasks typically considered for TBoxes are the following:

– *Satisfiability* checks whether a class C can have instances according to the current ontology.
– *Subsumption* checks whether a class D subsumes a class C according to the current ontology. Property subsumption is defined analogously.

As a representative reasoning task we consider *classification* of the ontology, *i.e.* computing the complete subsumption hierarchy of the ontology.

ABox reasoning tasks. ABox reasoning tasks usually come into play at runtime of the ontology. Reasoning tasks typically considered for ABoxes are the following:

– *Consistency* checks whether the ABox is consistent with respect to the TBox.
– *Instance checking* checks whether an assertion is entailed by the ABox.
– *Retrieval problem* retrieves all individuals that instantiate a class C, dually we can find all named classes C that an individual a belongs to.
– *Property fillers* retrieves, given a property R and an individual i, all individuals x which are related with i via R. Similarly we can retrieve the set of all named properties R between two individuals i and j, ask whether the pair (i, j) is a filler of P or ask for all pairs (i, j) that are a filler of P.

– *Conjunctive Queries* are a popular query formalism capable of expressing the class of selection/projection/join/renaming relational queries.

As ABox reasoning task we focus on answering conjunctive queries, as the vast majority of query languages models used in practice fall into this fragment and conjunctive queries have been found useful in diverse practical applications.

4.2 Performance Measures

Our primary performance measure is response time, *i.e.* the time that is needed to solve the given reasoning task. This means that we ignore the utilization of system resources, which could be another interesting measure for performance.

In our benchmarks we separate load time and query time to fairly compare the performance of the reasoners w.r.t. the different test ontologies:

– *Load Time* (P): Includes the time to do some important preparation before querying, *e.g.* load ontologies and check ABox consistency.
– *Response Time* (Q): Starts with executing the query and ends when all the query results were stored into a local variable. Usually, the query time means when a query is executed while not including the time for iterating the results.

4.3 Datasets and Queries

For the language fragments identified in Section 3, we chose a representative ontology for each fragment. The ontologies were chosen for two reasons. Firstly, they are popular, well established ontologies, which have been used in previous benchmarks. Secondly, they represent the cluster of ontologies as identified in section 3 in terms of size and ontological features used.

For each ontology, we used different datasets with increasing ABox size. Apart from one ontology (LUBM), which comes with its own ABox generator, the datasets are generated by duplicating originally existing ABox axioms for several times by renaming the individuals in these axioms. More details about these data sets can be found in Table 1.

For each ontology we used test queries that were either adopted from previous benchmarks, or explicitly defined for this benchmark. In particular we focused on conjunctive queries, as these are crucial in terms of complexity and response time.

VICODI. As a representative of the RDFS(DL) fragment, we used the VICODI ontology[6], and the following two ABox queries adopted from Motik and Sattler [11]:

$$Q_{v_1}(x) \equiv \mathit{Individual}(x) \tag{1}$$

$$Q_{v_2}(x,\, y,\, z) \equiv \mathit{Military\text{-}Person}(x),\ \mathit{hasRole}(y,\, x),\ \mathit{related}(x,\, z) \tag{2}$$

[6] `http://www.vicodi.org`

Table 1. Statistics of test ontologies.

Ontology	Class	Prop.	SubCl.	Equi.	SubPr.	Domain	Range	Functional	C(a)	R(a,b)	Axioms_total
vicodi_0									16942	36711	53876
vicodi_1									33884	73422	107529
vicodi_2	194	10	193	0	9	10	10	0	50826	110133	161182
vicodi_3									67768	146844	214835
vicodi_4									84710	183555	268488
swrc_0									4124	13712	27227
swrc_1									8248	27424	54328
swrc_2									12372	41136	81429
swrc_3									16496	54848	108530
swrc_4									20620	68560	135631
swrc_5	55	41	115	0	0	0	1	0	24744	82272	162732
swrc_6									28868	95984	189833
swrc_7									32992	109696	216934
swrc_8									37116	123408	244035
swrc_9									41240	137120	271136
swrc_10									45364	150832	298237
lubm_1									18128	49336	100637
lubm_2									40508	113463	230155
lubm_3	43	25	36	6	5	25	18	0	58897	166682	337221
lubm_4									83200	236514	477878
wine_0									247	246	719
wine_1									741	738	1721
wine_2									1235	1230	2723
wine_3									1729	1722	3725
wine_4									2223	2214	4727
wine_5	141	13	126	61	5	6	9	6	2717	2706	5729
wine_6									5187	5166	10739
wine_7									10127	10086	20759
wine_8									20007	19926	40799
wine_9									39767	39606	80879
wine_10									79287	78966	161039

SWRC. As a representative for the DLP fragment, we used the Semantic Web for Research Communities (SWRC) ontology[7]. The ontology is clearly settled above the RDFS(DL) fragment (in terms of expressiveness), since it contains universal quantification. However, this occurs only in the superclass description of class expressions, which keeps the ontology in the DLP fragment [6]. The following queries have been used for our benchmark:

$$Q_{s_1}(x) \equiv \mathit{PhDStudent}(x) \tag{3}$$

$$\begin{aligned} Q_{s_1}(x,\, y) \equiv\ & \mathit{ResearchTopic}(x),\ \ \mathit{isWorkedOnBy}(x,\, \text{“id2042instance”}), \\ & \mathit{dealtWithIn}(x,\, y) \end{aligned} \tag{4}$$

LUBM. The Lehigh University Benchmark (LUBM)[8] [7] was explicitly designed for OWL benchmarks. It models a scenario of the university domain and comes with its own ABox generator and a set of queries. Due to existencial restriction on the right side of class expressions, the LUBM ontology is in OWL Lite and just beyond the DLP fragment. We used the following three queries out of the LUBM queries:

$$Q_{l_1}(x,\, y_1,\, y_2,\, y_3) \equiv \mathit{Professor}(x),\ \ \mathit{worksFor}(x,\, \text{“University0.edu”}), \tag{5}$$

[7] `http://ontoware.org/projects/swrc/`

[8] `http://swat.cse.lehigh.edu/projects/lubm/index.htm`

$$\mathit{mastersDegreeFrom}(x,\ y_1),\ \mathit{teacherOf}(x,\ y_3),$$
$$\mathit{undergraduateDegreeFrom}(x,\ y_2)$$

$$Q_{l_2}(x) \equiv \mathit{Person}(x),\ \mathit{memberOf}(x,\ \text{“University0.edu”}) \tag{6}$$

$$Q_{l_3}(x,\ z) \equiv \mathit{Student}(x),\ \mathit{Department}(y),\ \mathit{memberOf}(x,\ y),\ \mathit{subOrganizationOf}(y,\ \text{“University0.edu”}) \tag{7}$$

Wine. The Wine ontology[9] is a prominent example of an OWL DL ontology. Since some reasoners are not able to handle nominals, we used the same datasets for the Wine ontology, as in previous benchmarks (cf. [11]), where nominals have been removed. We defined the following three ABox queries:

$$Q_{w_1}(x) \equiv \mathit{SemillonOrSauvignonBlanc}(x) \tag{8}$$

$$Q_{w_2}(x) \equiv \mathit{DessertWine}(x),\ \mathit{locatedIn}(x,\ \text{“GermanyRegion”}) \tag{9}$$

$$Q_{w_3}(x) \equiv \mathit{hasFlavor}(x,\ \text{“Strong”}),\ \mathit{hasSugar}(x,\ \text{“Dry”}),\ \mathit{locatedIn}(x,\ \text{“NewZealandRegion”}) \tag{10}$$

4.4 Discussion

We built our benchmark on several requirements, motivated by the work of Weithöner *et al.* [15]. Firstly, we distinguish between load time and response time. This distinction is necessary to demonstrate strengths and weaknesses of reasoners, that follow different paradigms, in particular materialization of inferred ABox assertions in the setup stage (*e.g.* Sesame, OWLIM). Our benchmark demonstrates how these approaches work on fairly simple ontologies (w.r.t. TBox complexity), as well as more complex ontologies, which these reasoners are unable to load at all. Secondly, we evaluate these measurements w.r.t. differently scaled ABoxes, but constant TBox. By doing so, we picked up methods, that have been used in previous benchmarks, which scale up the ABox by duplicating existing ABox assertions. While we evaluated reasoning tasks on four different expressivity (complexity) classes regarding TBox complexity, we did not put any effort in increasing TBox complexity for given ontologies. We rather focused on representative ontologies of given complexity for different classes of ontologies. We used only native interfaces of the different reasoners, and did not consider higher abstraction layers such as DIG. The influence of different interfaces is negligible anyway for querying large ontologies [15]. We did not evaluate cache influence or ABox changes on consecutive query requests, which goes beyond the scope of this paper. We also did not consider different serializations of ontologies, as we focus on well established ontologies, that do not occur in different serializations.

[9] `http://www.schemaweb.info/schema/SchemaDetails.aspx?id=62`

Table 2. Overview of Reasoners

Fragment	RDFS(DL)	DLP	OWL Lite	OWL DL
Example	VICODI	SWRC	LUBM	Wine
Sesame	×			
OWLIM	×	×		
KAON2	×	×	×	×[a]
HermiT	×	×	×	×
RacerPro	×	×	×	×[a]
Pellet	×	×	×	×

[a] Except for nominals

5 Overview of Reasoners

In this section we provide a short overview of the reasoners we used for our evaluations. Roughly, the reasoners can be grouped according to the employed reasoning techniques into three groups: In the first class of traditional DL reasoners (*e.g.* RacerPro, Pellet), tableau based algorithms are used to implement the inference calculus. A second alternative relies on the reuse of the techniques of deductive databases, based on a transformation of an OWL ontology into a disjunctive datalog program and to the utilization of a disjunctive datalog engine for reasoning as implemented in KAON2. A final class of reasoners – including Sesame and OWLIM – use standard rule engine to reason with OWL. Often the consequences are materialized when the ontology is loaded. However, this procedure is in principle limited to less expressive language fragments.

Table 2 shows an overview of the reasoners along with the language fragments they support.

5.1 Sesame

Sesame[10] is an open source repository for storing and querying RDF and RDFS information. OWL ontologies are simply treated on the level of RDF graphs. Sesame enables the connection to DBMS (currently MySQL, PostgreSQL and Oracle) through the SAIL (Storage and Inference Layer) module, and also offers a very efficient direct-to-disk SAIL called Native SAIL, which we used for our experiments. Sesame provides RDFS inferencing and allows querying through SeRQL, RQL, RDQL and SPARQL. Via the SAIL it is also possible to extend the inferencing capabilities of the system.

5.2 OWLIM

OWLIM is semantic repository and reasoner, packaged as a SAIL for the Sesame RDF database. OWLIM uses the TRREE engine to perform RDFS, and OWL DLP reasoning. It performs forward-chaining of entailment rules on top of RDF

[10] `http://openrdf.org`

graphs and employs a reasoning strategy, which can be described as total materialization. OWLIM offers configurable reasoning support and performance. In the "standard" version of OWLIM (referred to as SwiftOWLIM) reasoning and query evaluation are performed in-memory, while a reliable persistence strategy assures data preservation, consistency and integrity.

5.3 KAON2

KAON2 is a free (free for non-commercial usage) Java reasoner for $\mathcal{SHIQ}$ extended with the DL-safe fragment of SWRL. Contrary to most currently available DL reasoners does not implement the tableau calculus. Rather, reasoning in KAON2 is implemented by novel algorithms which reduce a $\mathcal{SHIQ}(D)$ knowledge base to a disjunctive datalog program. These novel algorithms allow applying well-known deductive database techniques, such as magic sets or join-order optimizations, to DL reasoning

5.4 HermiT

HermiT is a freely available theorem prover for description logics. The reasoner currently fully handles the DL $\mathcal{SHIQ}$. The support for $\mathcal{SHOIQ}$ is currently being worked on. The main supported inference is the computation of the subsumption hierarchy. HermiT can also compute the partial order of classes occurring in an ontology. HermiT implements a novel hypertableau reasoning algorithm. The main aspect of this algorithm is that it is much less non-deterministic than the existing tableau algorithms. A description of the reasoning technique using hypertableaux for $\mathcal{SHIQ}$ can be found in [12].

5.5 RacerPro

The RacerPro system [8] is an optimized tableau reasoner for $\mathcal{SHIQ}(D)$. For concrete domains, it supports integers and real numbers, as well as various polynomial equations over those, and strings with equality checks. It can handle several TBoxes and several ABoxes and treats individuals under the unique name assumption. Besides basic reasoning tasks, such as satisfiability and subsumption, it offers ABox querying based on the nRQL optimizations. It is implemented in the Common Lisp programming language. Recently, RacerPro has been turned into the commercial (free trials and research licenses available) RacerPro[11] system, which we used for our experiments.

5.6 Pellet

Pellet[12] [13] is a free open-source Java-based reasoner for $\mathcal{SROIQ}$ with simple data types (*i.e.* for OWL 1.1). It implements a tableau based decision procedure

[11] `http://www.RacerPro-systems.com/`

[12] `http://www.mindswap.org/2003/pellet/index.shtml`

Table 3. Performance results of classification.

KB	vicodi[a]	swrc	lubm	wine
Sesame(P)	0.769	0.635	0.467	1.180
Sesame(Q)	0.099	-	-	-
OWLIM(P)	0.580	12.990	0.684	0.964
OWLIM(Q)	0.071	0.079	0.093[b]	-
KAON2(P)	0.387	0.383	0.349	0.553
KAON2(Q)	2.746	1.137	1.010	5.141
HermiT(P)	0.889	0.776	0.708	1.090
HermiT(Q)	0.180	0.046	0.046	0.465
RacerPro(P)	0.110	0.092	0.072	0.168
RacerPro(Q)	0.080	0.056	0.356	0.607
Pellet(P)	0.563	0.518	0.404	0.835
Pellet(Q)	1.145	0.346	0.253	9.252

[a] The fact, that reasoners take longer to classify VICODI than SWRC despite higher expressivity of SWRC is due to the larger number of classes in VICODI (cf. Table 1).

[b] Even though OWLIM is not able to process OWL Lite ontologies in general, it is able to process the particular set of features LUBM uses.

for general TBoxes (subsumption, satisfiability, and classification) and ABoxes (retrieval, conjunctive query answering). Pellet employs many of the optimizations for standard DL reasoning as other state-of-the-art DL reasoners. It directly supports entailment checks and optimised ABox querying through its interface.

6 Experiments

In this section we report on the benchmarking experiments performed. A technical report describing the full results of the experiments is available at [1].

Our tests were performed on a Linux 2.6.16.1 System. The Sun Java^{TM} 1.5.0 Update 6 was used for Java-based tools and the maximum heap space was set to 800 MB. For each reasoning task, the time-out period was assigned 5 minutes.

For each reasoning task, a new instance of the reasoner is created and the test ontology is loaded. No methods about optimization for the reasoners are called, using default settings.

Classification. We consider classification as a representative task for TBox reasoning. Table 3 compares the results for classification among the pure TBoxes of the four ontologies selected for our tests, where 'P' indicates load time and 'Q' means classification time.

Regarding the classification experiment, it can clearly be observed, that RacerPro outperforms all other systems in terms of load time, while HermiT performs best in terms of classification time for all test ontologies, but VICODI. It should be noted, that despite best performance in actual classification, HermiT is about one order of magnitude slower in loading the ontologies than RacerPro.

The distinction between load time and classification time becomes important if applications frequently operate on preloaded ontologies, where load time can be neglected. In this case, HermiT should be the reasoner of choice, as described in the previous paragraph.

For those TBox reasoning tasks where the ontology has to be re-loaded for each re-classification (*e.g.* loose coupling between ontology development tools and reasoners), the distinction between load time and classification time becomes less important. By disregarding this distinction in the classification experiment and considering the total time only, RacerPro performs best in all expressivity fragments. Another observation is that lightweight reasoning and storage systems such as Sesame and OWLIM do not bring any advantage in expressivity fragments they are tailored to. Indeed, they are still outperformed by RacerPro and perform only slightly better than HermiT and Pellet.

Summing this up, the observation is, that tableau based systems generally outperform the resolution based KAON2 for TBox reasoning tasks. If load time is of minor importance, HermiT with its novel hypertableau method performs best in classifying ontologies, apart from lightweight VICODI, where OWLIM still performs better. If caching of preloaded ontologies is not possible and has to be done for any (re-)classification, RacerPro clearly performs best w.r.t. total execution time.

Conjunctive Queries. Figure 3 illustrates the results of the conjunctive query experiments[13]. W.r.t. RDFS(DL) ontologies, apparently there is a clear trade-off between loading and query time as Sesame is the slowest system to load but the fastest system to respond. For the VICODI datasets, the average query time ranges from 0.18 to 0.33 seconds for Sesame.

Actually OWLIM performs slightly worse than Sesame in responding for RDFS(DL) ontologies and query time varies from 0.27 to 0.58 seconds. The time to load shows much better results than Sesame. From the overall view, OWLIM has better performance than Sesame. Considering the OWL DLP ontology SWRC, OWLIM is the fastest reasoner on query time which varies from 0.06 to 0.12 seconds.

Obviously, KAON2 is the best system w.r.t the overall performance to load and respond, and shows favorable scalability. Although Sesame and OWLIM display good performance for the datasets they support, KAON2 is just slightly slower but much faster to load. Pellet also responds faster than KAON2 in most cases for expressive ontologies (except for RDFS(DL) ontologies). However, KAON2 is much faster in loading and more scalable than Pellet which produces time-outs for Wine ontologies larger than wine_5. RacerPro responds slightly faster for small Wine ontologies but is significantly less scalable than KAON2, resulting in time-outs for the largest ontologies, such as vicodi_4 and Wine datasets from wine_5. The performance of Pellet lags behind and produces time-outs as soon as the ABox reaches medium size.

[13] At the time of this evaluation, conjunctive query answering was not implemented for HermiT yet. For that reason we used HermiT only in the classification experiment.

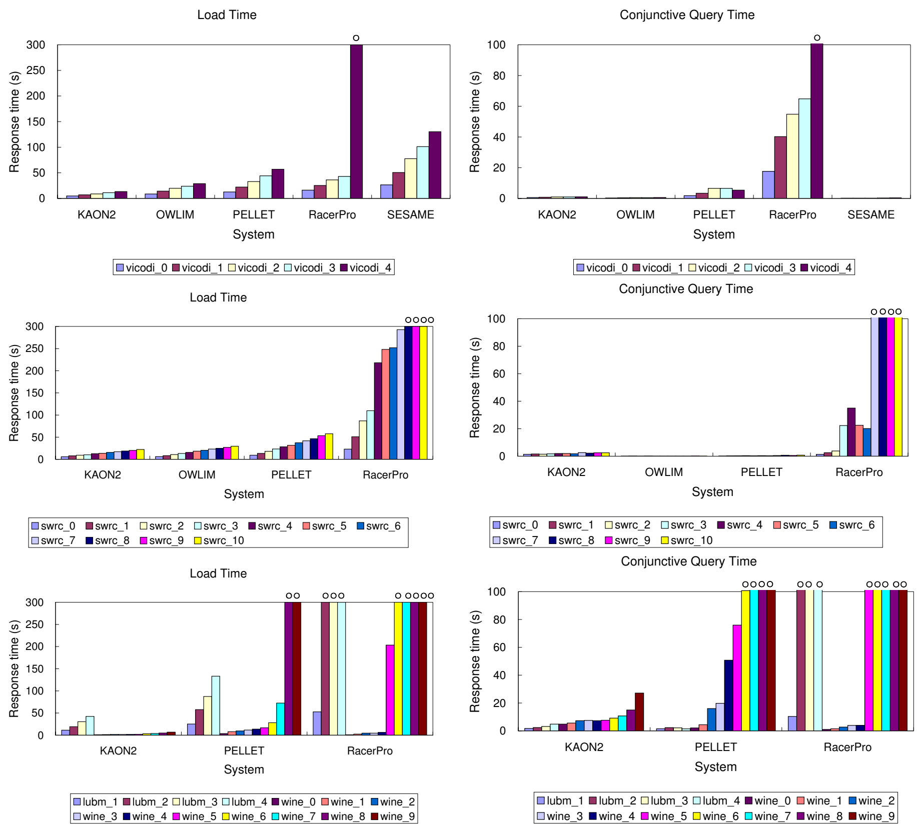

Fig. 3. The performance of conjunctive queries for all datasets. The figures show the average load and query time of all queries with a particular reasoner on a particular dataset. "O" on top of a bar indicates time-out.

For ABox reasoning tasks, in particular conjunctive query answering, the choice of the reasoner depends on the expressivity of the ontology. Whereas lightweight ontologies – in our case VICODI and SWRC for RDFS(DL) and OWL DLP rsp. – can be materialized at the loading stage by Sesame and OWLIM, they are unable to do so for ontologies of higher expressivity. In particular OWLIM performed very well, while still being able to process OWL DLP, and hence should be the choice for ABox reasoning with lightweight ontologies. For the more expressive language fragments, OWL Lite and OWL DL, KAON2 as resolution based reasoner outperforms the tableau based methods Pellet and RacerPro. In fact, KAON2 is the only system that is able to answer queries for all LUBM and Wine ontologies within the given time range of 5 minutes.

Figure 4 depicts a differentiated case along two main dimensions: language complexity and ABox size. While RacerPro is the system of chose in settings with

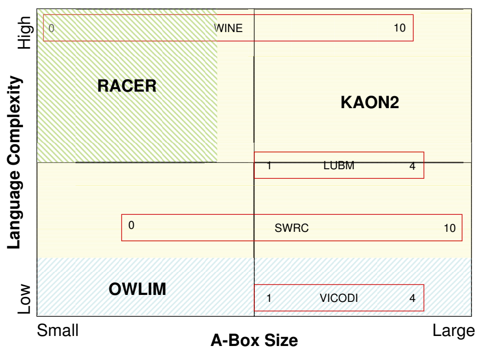

Fig. 4. Recommendation for reasoner selection

high complexity and small ABoxes, OWLIM can be generally recommended for low complexity settings while KAON2 is the best alternative for all other cases.

7 Conclusion

In today's landscape of ontologies, we observe a wide spectrum of ontologies that differ in terms of language expressivity, as well as their complexity in terms of their size of TBox and ABox. A number of rather different reasoning techniques are implemented in state-of-the-art OWL reasoners. In our benchmarks we have shown that it is important to understand the strengths and weaknesses of the different approaches in order to select an adequate reasoner for a given reasoning task. It does not come as a surprise that there is no clear "winner" that performs well for all types of ontologies and reasoning tasks.

As general conclusions we can summarize our results in that (1) reasoners that employ a simple rule engine scale very well for large ABoxes, but are in principle very limited to lightweight language fragments, (2) classical tableau reasoners scale well for complex TBox reasoning tasks, but are limited with respect to their support for large ABoxes, and (3) the reasoning techniques based on reduction to disjunctive datalog as implemented in KAON2 scale well for large ABoxes, while at the same time they support are rich language fragment. If nominals are important for a given scenario, Pellet is the only reasoner in this benchmark, which has adequate support.

An important current research topic is the investigation of tractable fragments of the OWL language [5, 10] and the development of reasoners specialized for these fragments. During our analysis of the ontology landscape we identified many of the ontologies on the web as erroneous, which hampers a satisfying expressivity analysis. As future work, we will extend our analysis taking new fragments into account, as well as providing means to reveal more detailed statistics of the ontology landscape on the web.

References

1. J. Bock, P. Haase, Q. Ji, and R. Volz. Benchmarking OWL Reasoners - Technical Report. Technical report, University of Karlsruhe, 2007. `http://www.aifb.uni-karlsruhe.de/WBS/pha/publications/owlbenchmark.pdf`.
2. I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors. *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*. Springer, 2006.
3. M. d'Aquin, C. Baldassarre, L. Gridinoc, S. Angeletou, M. Sabou, and E. Motta. Characterizing knowledge on the semantic web with WATSON. In *Proceedings of the 5th International EON Workshop, International Semantic Web Conference (ISWC 2007)*, Buasn, Korea, 2007.
4. T. Gardiner, D. Tsarkov, and I. Horrocks. Framework for an automated comparison of description logic reasoners. In Cruz et al. [2], pages 654–667.
5. B. Cuenca Grau. Tractable Fragments of the OWL 1.1 Web Ontology Language, February 2006. `http://owl-workshop.man.ac.uk/Tractable.html`, accessed 22/06/2007.
6. B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logics. In *Proc. of WWW 2003, Budapest, Hungary, May 2003*, pages 48–57. ACM, 2003.
7. Y. Guo, Z. Pan, and J. Heflin. Lubm: A benchmark for owl knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.
8. V. Haarslev, R. Möller, and M. Wessel. Querying the semantic web with racer + nrql. In *Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04), Ulm, Germany, September 24*, 2004.
9. C. M. Keet and M. Rodríguez. Comprehensiveness versus scalability: guidelines for choosing an appropriate knowledge representation language for bio-ontologies. Technical Report KRDB07-5, Faculty of Computer Science, Free University of Bozen-Bolzano, 2007.
10. M. Krötzsch, S. Rudolph, and P. Hitzler. Complexity boundaries for horn description logics. In *Proceedings of the 22nd AAAI Conference on Artficial Intelligence*, pages 452–457, Vancouver, British Columbia, Canada, 2007. AAAI Press.
11. B. Motik and U. Sattler. A comparison of reasoning techniques for querying large description logic aboxes. In M. Hermann and A. Voronkov, editors, *LPAR*, volume 4246 of *Lecture Notes in Computer Science*, pages 227–241. Springer, 2006.
12. B. Motik, R. Shearer, and I. Horrocks. Optimized reasoning in description logics using hypertableaux. In F. Pfenning, editor, *CADE*, volume 4603 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 2007.
13. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A Practical OWL-DL Reasoner. Technical report, University of Maryland Institute for Advanced Computer Studies (UMIACS), 2005. `http://mindswap.org/papers/PelletDemo.pdf`.
14. T. D. Wang, B. Parsia, and J. A. Hendler. A survey of the web ontology landscape. In Cruz et al. [2], pages 682–694.
15. T. Weithöner, T. Liebig, M. Luther, S. Böhm, F. von Henke, and O. Noppens. Real-world Reasoning with OWL. In E. Franconi, M. Kifer, and W. May, editors, *Proceedings of the European Semantic Web Conference, ESWC2007*, volume 4519 of *Lecture Notes in Computer Science*. Springer-Verlag, July 2007.