Fourth International Conference on Semantics, Knowledge and Grid

A Testing Framework for OWL-DL Reasoning

Marian Babik, Ladislav Hluchy

[#]*Department of Parallel and Distributed Computing,*
Institute of Informatics, Slovak Academy of Sciences
Dubravska cesta 9, 84507 Bratislava, Slovakia
[1]Marian.Babik@saske.sk
[2]Ladislav.Hluchy@savba.sk

***Abstract*— OWL and RDF/RDFS are ontological languages developed by the World Wide Web Consortium (W3C), which have become a de facto standard for the ontological descriptions in various domains. The evolution of these standards was influenced by the numerous advances in the research of knowledge representation and reasoning. Although support for reasoning and standardized representation is the key benefit of these technologies, there is a lack of existing test frameworks, which would be capable of addressing many crucial aspects of the Semantic Web applications.**

In this paper we propose a methodology for automated testing of OWL reasoners based on the real-world ontologies. This specification covers both terminological and assertional reasoning as well as checking of the correctness of the answers. An open-source implementation of such framework is described and a study of initial results is provided. The tests cover an extensive set of reasoners and ontologies and provide a state-of-the-art insight into the field of OWL reasoning.

I. Introduction

Semantic Web is an established research domain, which tries to extend the current Web technologies by providing a well defined meaning to the services and information [1]. It has provided numerous contributions in knowledge representation and reasoning and it is seen as a possible infrastructure, which can provide an environment for hosting and managing heterogeneous data and services. The use of the Semantic Web technologies in data integration and schema-based peer-to-peer systems is not novel. It has already been useful in providing a scalable solutions for the distributed computing (Semantic Grid), web services (Semantic Web Services), search and retrieval in peer-to-peer systems [2], [3].

OWL and RDF/RDFS are ontological languages developed by the World Wide Web Consortium (W3C) [4], [5]. Although initially specified by the requirements of the Semantic Web community, they have become a standard for ontological descriptions in several domains including earth sciences, bioinformatics, chemistry, astronomy, aerospace and the automotive industries [6]. Both languages are based either on the first-order logic or on its subset, i.e. description logics. Therefore the evolution of the standards have been heavily influenced by the research in the representation and reasoning of these logics. The standardization also led to the development of numerous tools, e.g. Protege, Swoop[1], and reasoners such as Pellet, Racer-Pro, KAON2, etc [7], [8], [9].

[1]http://protege.stanford.edu/, http://www.mindswap.org/2004/SWOOP

One of the key benefits of the Semantic Web technologies is to support reasoning and standardized representation of heterogeneous data, thus enabling Web-scale *interoperability* [10]. This, however, requires not only the specification of standards, but also a proper reasoning methods, which can compute consistency and derive the same set of facts given the same set of ontologies. One of the goals of the standardization is that this process is independent of the chosen reasoner. However, the existing testing suites are rarely capable of addressing such goal, since they mostly rely on testing a particular feature of the reasoner or supported language. While such tests are important for checking the correctness of the reasoner implementation, they do not address a more urgent requirement, i.e. reasoning with datasets based on real-world ontologies.

Once a correctness of the reasoners can be confirmed, it is desirable to compare the performance of various reasoners. This is necessary since the intractability of the current ontological reasoning leads to many optimizations focused on the patterns of particular real-world ontologies. Such heuristics can often play a key role in performance of the reasoner. Various existing testing frameworks are trying to compare performance of the reasoners, but often those are limited to a particular set of ontologies, features or reasoning tasks (e.g. ABox or TBox reasoning).

Another, more recent development in the Semantic Web is trying to focus on the aspects of web-scale reasoning, i.e. combining information retrieval and reasoning. In such context it is very difficult to assume that soundness and completeness of reasoning can be accomplished. This means that the current set of standards and reasoning methods will be further enriched to capture such requirements. Currently, however there are no existing test frameworks which would allow a comparison of reasoners combining information retrieval and ontological reasoning.

The contribution of this paper is in defining a methodology for extensible testing framework for both assertional (Abox) and terminological (TBox) reasoning in OWL and specification of a dataset of real-world ontologies that can cover many aspects of the ontological languages. An open-source implementation of such testing framework is described and the initial results of the tests are presented and discussed. The paper concludes with a plan for the future work.

978-0-7695-3401-5/08 $25.00 © 2008 IEEE
DOI 10.1109/SKG.2008.14

II. METHODOLOGY

A methodology for the automated testing of the terminological and assertional reasoning is based on the foundations of description logic, which can be easily transformed to a relevant ontological language such as OWL. The methodology is described as a set of steps that provide a complete testing procedure for ontological reasoning.

Initially, a set of ontologies (URIs) forming a testing dataset (DS) is specified. This dataset is then preprocessed in order to extract the relevant meta-data from the ontologies. The extracted meta-data provides the information about the expressivity of the ontology (i.e. type of logical constructs used), statistics of the language features (number of concepts, axioms, relations, individuals, nominals, etc.) and classification of the hierarchical patterns presented in the ontology (structures of concept or role hierarchies, e.g. lists, trees, graphs). Such meta-data are important to understand the results of the tests as they provide valuable insight into possible causes of strengths or weaknesses of tested reasoners. Further, the metadata can be mined to extract the patterns of the existing real-world ontologies and provide a basis for generating more complex datasets. Such datasets are especially important for the theoretical foundations of the combined information retrieval and ontological reasoning as current dataset are quite limited in size and numbers.

The main function of the testing framework is to determine the time for key reasoning tasks for each ontology and each reasoner (R). The information includes time needed for loading the ontology, perform certain queries and also the actual query returned by the reasoner. Different reasoners use different reasoning methods to answer queries and perform caching and indexing at different stages of the answering. Therefore we have developed a main loop consisting of several standard queries, for each measuring elapsed time. This allows us to determine where different reasoners spent most the time, but also to compare performance of different reasoner based on overall results.

Algorithm 1: Testing algorithm

```
Data: DS, R
Result: t_i
forall KB ∈ DS do
    forall r ∈ R do
        t_1 ← time(load_r(KB))
        t_2 ← time(∀C_i ∈_r KB)
        t_3 ← time(∀P_i ∈_r KB)
        t_4 ← time(⊤ ⊨_r KB)
        forall C_i do
            t_5 ← time(C_i ⊨_r KB)
        end
        forall C, D ∈_r KB, whereC ≠ D do
            t_6 ← time(C ∩ ¬D ⊨_r KB)
        end
        forall C_i do
            t_7 ← time(C(a) ⊨_r KB)
        end
        forall R_i do
            t_8 ← time(P(a,b) ⊨_r KB)
        end
    end
end
```

The main loop of the testing is shown in Alg. 1. Input to the algorithm is a set of reasoners (R) and ontologies (URIs). Initially, the ontology is loaded into the reasoner by performing deserialization and axiomatization, resulting in a description logic knowledge base KB (t_1). The knowledge based consists of classes (C), properties (P) and their instances ($C(a), P(a,b)$). Then reasoner is queried for all the named classes (t_2) and properties (t_3). After initial checks a set of terminological queries (schema-based queries) are performed. First query checks the consistency of the ontology by checking the satisfiability of the $\top$ class also known as owl:Thing (t_4). Then we query the reasoner for satisfiability of all known concepts in the ontology (t_5) and finally we perform the classification of the ontology, i.e. determining the concept hierarchy - parents and children of all named classes (t_6). The set of assertional queries then checks for the instances of given classes (t_7) and properties (t_8), i.e. instance-based queries. Generally, each named class or property of the KB can be queried for its instances, but in case of a benchmark ontology (e.g LUBM) we have followed a set of conjunctive benchmark queries.

Each reasoner is terminated and restarted before loading any ontology. This ensures that every reasoner is in consistent state before starting the tests. Also since some of the reasoners fail to respond in an upper time limit (due to memory leak or other implementation issues), it is necessary to restart them in order to continue with the rest of the tests. The individual queries are timed, which gives interesting insight on where the reasoner spends most of the time. The total time is computed for both terminological and assertional reasoning steps, but the overall time for the complete test is probably of most interest. The benchmarking process is fully automated and thus can be run periodically giving opportunity for the developers to check and improve their implementations. All the results and times are recorded in a data storage allowing convenient access for user to view and analyze the data.

Each test can end in three different ways. It can complete successfully and store the results and times in the data storage. Or it can fail either due to lack of resources (time or memory) or inability to load or process the given ontology. Successful test are then considered in the evaluation of the correctness of the reasoner.

The correctness of the reasoner is an important measure as it determines whether the implemented decision procedure (reasoning) is sound and complete, i.e. it return correct responses wrt. underlying logic theory. Incomplete answers are usually computed faster, thus providing a better overall performance of the reasoner. Therefore, detecting incomplete or unsound reasoning is important in promoting fairness among performance comparisons. The usual way of determining correctness of the reasoner is to have a small artificially computed ontology for which it is possible to compute and check the inferences manually. This is however not possible in our setting as the actual size of ontologies can be potentially very large.

Since it is impossible to determine the correctness of reasoning in our settings, we rely on the comparison of results

between the reasoners to approximate such evaluation. This assumes that consistency among multiple reasoners implies a high probability of correctness. This is also supported by the fact that we are testing independently developed reasoners, some of which rely on completely different logical theories, e.g. tableau, disjunctive datalog, hypertableau [11], [12], [9].

III. Implementation

The implementation is based on a plugin architecture allowing free extensibility for adding new reasoners and ontologies, including complete benchmark sets such as LUBM[2]. The core of the system is a test controller, which instantiates reasoners and ontologies based on a configuration. The configuration describes a set of global properties, particular reasoners, ontologies and queries that should be run as part of the test.

The global properties configure access to the data storage (RDBMS) as well as setup of the basic directories (location of reasoner APIs, ontologies, outputs, etc.). The configuration is based on the Java configuration API.

Since the system is written in Java it mainly interfaces with the reasoner either directly or through a java-based APIs specified by the non-java reasoners. We did not use Description Logic Interface (DIG)[3] due to many implementation issues, performance overhead and inability to support assertional queries. A direct java access to reasoners supports the best possible performance for many reasoners, as Java is a common language to many Semantic Web applications. However, we plan to also integrate DIG 2.0 in the future as it provides a standardized common API. The system uses standard Java interfaces for parsing ontologies and accessing reasoners, i.e. OWL or Jena API and SWOOP preprocessing capabilities.

The system can process ontologies in RDF and OWL languages. The set of OWL sublanguages is restricted to OWL-Lite and OWL-DL, since OWL-Full is undecidable and thus irrelevant for our tests. The terminological queries are handled with OWL API and the assertional queries are based on the SPARQL and SPARQL APIs of the corresponding reasoners.

We have started to work on a web-based interface for the testing framework, which could provide a very flexible setup with graphs, analysis and detailed statistics for every run.

IV. Evaluation

In this section we present a comparison of the performance of query answering of prominent OWL reasoners, which should provide an insight into a practical applicability of the existing approaches.

It should be noted that due to a large number of optimizations and high complexity of the methods, it is very difficult to separate the reasoning methods from their respective implementations. There are numerous low-level optimizations that are implemented in the methods and the choice of e.g. data structures or memory management can easily dominate the reasoning time. Furthermore, the implementations are written in different languages and usually provide only proprietary source code, which makes the evaluation even more complex. Therefore the results of this section presents an overview of the performance, which can be expected in the real-life scenarios, rather than a definitive measure of the complexity of the reasoning algorithms.

We have compared the performance of the four recent description logics reasoners, KAON2, Pellet, Racer, Fact and HermiT [9], [8], [7], [13], [11]. We did not consider the other description logics reasoners due to their limitations; DLP does not support ABox reasoning [14], LOOM is incomplete [15], CLASSIC, OWLIM and JENA[4] support only a subset of the SHIN(D) description logics (OWL-lite) [16], [17].

The sequence of API calls that we used for each reasoner were determined from the LUBM benchmarks. We have tried to accommodate all the recommendations for the performance evaluation of each reasoner. For each reasoning task we have started a fresh instance of the reasoner and loaded the test knowledge base. This was done mainly due to significant problems with memory management of the reasoners during repetitive querying [18]. We have measured the time required to execute each step. We have also assured that all systems returned the same answers.

In case of the tableau-based reasoners the optimizations of the ABox queries usually involve caching of computation results; thus the performance can increase with subsequent queries. Further, both Racer and Pellet check the ABox consistency during which they compute the index for the instance retrieval, which severely affects its initial performance. Since we have not considered any caching and materialization techniques in our approach, we have measured both the time for ABox consistency and the time to answer the query. It should be noted that computing the index for the instance retrieval is not feasible in many applications due to a large number of individuals. In case of KAON2 we have not measured the time to compute the datalog program as it was insignificant.

All tests were performed on the laptop computer (T60) with 1.8GHz memory and 1 GB of RAM, running Linux kernel 2.6.25-6. For Java-based reasoners we have used Java runtime 1.6.0 Update 6 with virtual memory restricted to 800 MB. We run each reasoning task five times and plotted the average of the set. Each task had a time limit of 5 minutes. Tests that either run out of memory or out of time are denoted with time 300000.

A. Test Ontologies

Initially we have based our test on over 50 real-world ontologies developed within the Semantic Web Community including well known benchmark ontologies such as Galen, LUBM, etc. This set has provided a very good mix of complex ABox and TBox reasoning. In order to obtain sufficient number of individuals we have performed ABox replication, i.e. duplication and renaming of the individuals in the ABox. Tables II,I show the statistics about the structure and complexity of the most prominent ontologies tested. These ontologies

[2]http://owland.sourceforge.net

[3]http://dig.cs.manchester.ac.uk/

[4]http://jena.sourceforge.net/

TABLE I

STATISTICS OF THE PROMINENT ONTOLOGIES WITH COMPLEX ABOX

KB	DL	$\|C\|$	$\|P\|$	$\|D\|$	$C(a)$	$P(a,b)$
semintec_1	SHIF	61	16	0	17941	47248
semintec_2					35882	94496
semintec_3					53823	141744
semintec_4					71764	188992
semintec_5					89705	236240
vicodi_1	ALHI	196	10	10	16942	36711
vicodi_2					33884	73422
vicodi_3					50826	110133
vicodi_4					67768	146844
lubm_1	SHI(D)	45	25	7	18128	49336
lubm_2					40508	113463
lubm_3					58897	166682
lubm_4					83200	236514
wine_8	SHIF	142	13	0	20007	19926
wine_9					39767	39606
wine_10					79287	78966
owls_1	SHIF(D)	93	64	1	50011	0
owls_2					75011	0
owls_3					120011	0
tree_5_4_7	ALC	1367	0	0	9548	0
tree_5_4_10					13640	0
tree_5_5_1					3905	0
tree_5_5_5		3908			19525	0
tree_5_5_7					27335	0
tree_5_5_10					39050	0

TABLE II

STATISTICS OF THE PROMINENT ONTOLOGIES WITH COMPLEX TBOX

KB	DL	$\|C\|$	$\|P\|$	$\|D\|$	$C(a)$	$P(a,b)$
dolce	SHIF(D)	125	253	2	0	0
galen simplified	SHIF	2749	261	0	0	0
galen original	SHIF	2750	413	0	0	0
NCI	ALC	27654	70	0	0	0

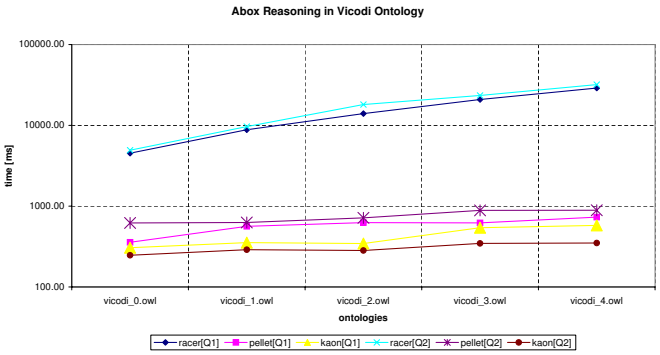
provide a very good mix of different expressivity of the $TBox$, while containing a large number of instances in the $ABox$.

B. Performance tests

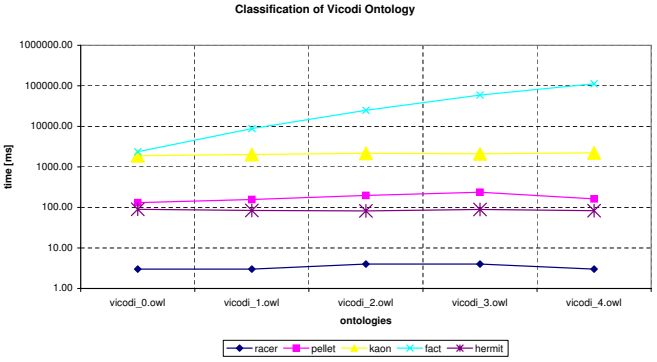
Vicodi. Since Vicodi ontology contains only very simple TBox, it can be expected that resolution-based decision procedure will dominate the test with increasing number of individuals. We have performed the following conjunctive queries over the Vicodi ontology:

$$Q_1(x,y,z) = MilitaryPerson(x), hasRole(x,y), related(x,z)$$

Further we have performed a classification of the TBox (Q_2). The results in Figure 1 show that Racer and HermiT are dominant in answering terminological query Q_2, while KAON2 performs better on the Q_1 queries. This is due to its connection with the deductive database, which plays a key role in answering conjunctive queries for a simple TBox. It can be seen that although Racer and HermiT employ different



ontology	racer[Q1]	pellet[Q1]	kaon[Q1]	racer[Q2]	pellet[Q2]	kaon[Q2]
vicodi_0.owl	4529.00	357.00	306.00	4924.00	620.00	247.00
vicodi_1.owl	8828.00	563.00	354.00	9669.00	628.00	289.00
vicodi_2.owl	13979.00	625.00	345.00	18102.00	716.00	283.00
vicodi_3.owl	20895.00	621.00	541.00	23424.00	887.00	346.00
vicodi_4.owl	28948.00	731.00	578.00	31919.00	890.00	350.00



ontology	racer	pellet	kaon	fact	hermit
vicodi_0.owl	3.00	131.00	1889.00	2369.00	90.00
vicodi_1.owl	3.00	157.00	2022.00	8803.00	84.00
vicodi_2.owl	4.00	197.00	2163.00	24911.00	82.00
vicodi_3.owl	4.00	237.00	2100.00	59237.00	89.00
vicodi_4.owl	3.00	162.00	2213.00	112254.00	83.00

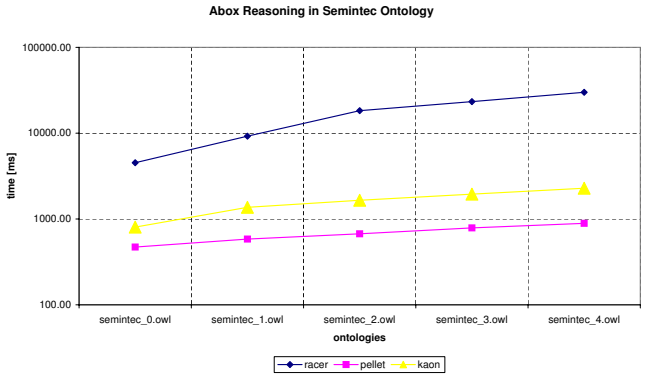
Fig. 1. Execution time for the VICODI ontology, Q_1 (left) and Q_2 (right)

strategies for answering terminological queries, there is only a small gap between their performances. This is mainly due to various different optimizations that were developed over the years for the Racer reasoner.

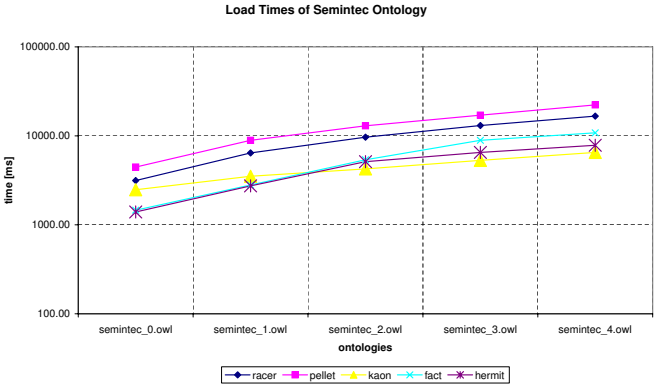
Semintec. Similar to the previous case, Semintec is also a very simple ontology thus, the results can be expected to follow the same pattern. Unlike VICODI, Semintec contains functional roles, which are more difficult for the deductive databases (KAON2). We have performed the following query:

$$Q_1(x,y,z) = Man(x), isCreditCardOf(y,x), Gold(y), livesIn(x,z), Region(z)$$

We have also performed classification as a query (Q_2). We have also measured a time it takes to load the ontology. The results are shown in Fig. 2. It can be seen that the performance of KAON2 deteriorates due to the functional roles. In conjunctive query answering Pellet outperforms both Racer and KAON2. In terms of the TBox classification the situation follows the previous tests with Vicodi. The only exception is performance of Fact, which shows quite large sensitivity to the number of instances. The load times for the

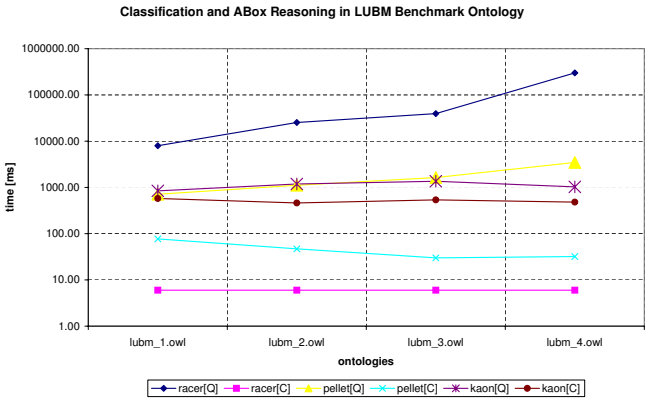


ontology	racer	pellet	kaon
semintec_0.owl	4527.00	471.00	804.00
semintec_1.owl	9229.00	583.00	1368.00
semintec_2.owl	18302.00	672.00	1654.00
semintec_3.owl	23333.00	788.00	1947.00
semintec_4.owl	29958.00	890.00	2285.00

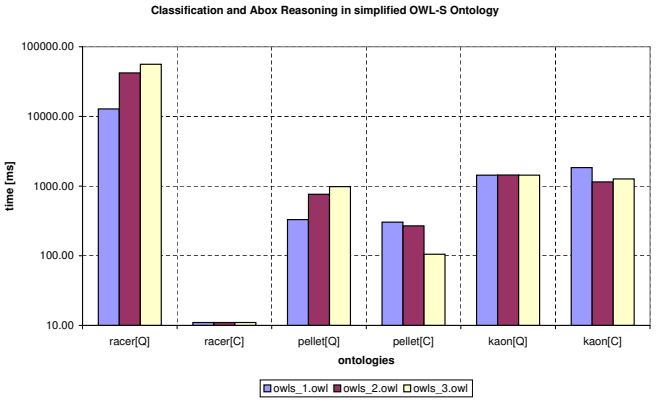


ontology	racer	pellet	kaon	fact	hermit
semintec_0.owl	3144.00	4461.00	2478.00	1458.00	1394.00
semintec_1.owl	6430.00	8907.00	3512.00	2805.00	2742.00
semintec_2.owl	9667.00	12963.00	4254.00	5387.00	5121.00
semintec_3.owl	13054.00	17089.00	5301.00	8903.00	6518.00
semintec_4.owl	16667.00	22272.00	6511.00	10826.00	7839.00

Fig. 2. Execution time for the SEMINTEC ontology for Q_2 (left) and load times (right)



ontology	racer[Q]	racer[C]	pellet[Q]	pellet[C]	kaon[Q]	kaon[C]
lubm_1.owl	7966.00	6.00	712.00	77.00	844.00	577.00
lubm_2.owl	25327.00	6.00	1113.00	47.00	1187.00	462.00
lubm_3.owl	39399.00	6.00	1637.00	30.00	1360.00	538.00
lubm_4.owl	300000.00	6.00	3464.00	32.00	1030.00	482.00



ontology	racer[Q]	racer[C]	pellet[Q]	pellet[C]	kaon[Q]	kaon[C]
owls_1.owl	12827.00	11.00	330.00	304.00	1436.00	1844.00
owls_2.owl	42154.00	11.00	764.00	269.00	1441.00	1148.00
owls_3.owl	56108.00	11.00	982.00	105.00	1435.00	1266.00

Fig. 3. Execution time for the LUBM benchmark (left) and OWL-S ontologies (right)

Semintec ontology closely follows the number of instances with very small gaps between the reasoners.

Lehigh University Benchmark (LUBM) is comparable to Semintec and VICODI in terms of size; however it contain more complex TBox concepts. Since the original benchmark contains several queries for which we have had similar results, we have chosen a set of simple and complex queries:

$$Q_1(x) = Chair(x)$$
$$Q_2(x, y, z) = Student(x), Faculty(y), Course(z), advisor(x, y), takesCourse(x, z), teacherOf(y, z)$$

A pairwise comparison of overall ABox and TBox reasoning is shown in Fig. 3. Both Fact and HermiT were unable to load and classify the ontology due to missing support for specific ontological expressivity. KAON2 and Pellet show the most stable performance for both terminological and assertional reasoning. Although Racer outperforms both KAON2 and Pellet in terminological reasoning, its performance deteriorates for assertional reasoning.

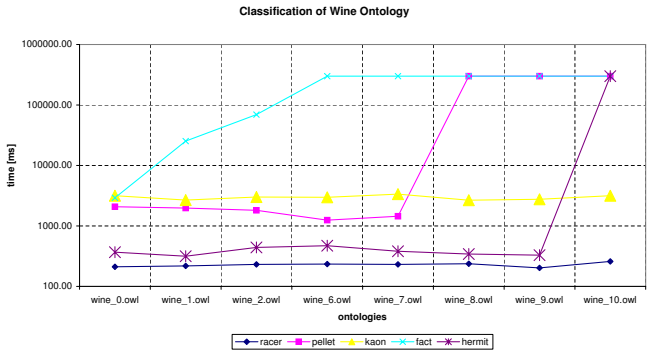
OWL-S provides one of the most frequently used ontologies in the field of semantic web services[5]. It contains a quite complicated TBox and can be connected to many other ontologies, which provide the domain model of the application. In our case we have performed the queries over a set of real-world ontologies developed within the project K-WfGrid [19]. Apart from classification (Q_2) we have performed the following query:

$$Q_1(x) = ServiceProfile(x)$$

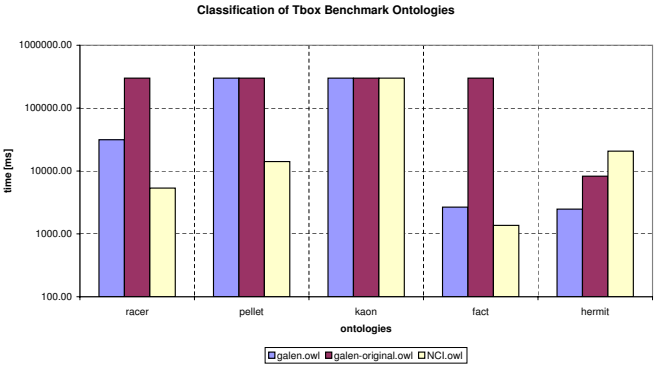
The results are shown in Fig. 3. Extracting the existing $ServiceProfiles$ is one of the most frequent queries in both composition and discovery of services, as both rely on the ontological model of the $ServiceProfile$. As in previous test with increased expressivity Pellet outperform the existing reasoners in both terminological and assertional reasoning.

Wine. Even more complex ontology than OWL-S is the Wine ontology, as it contains multiple disjunctions, which

[5]http://www.w3.org/Submission/OWL-S/

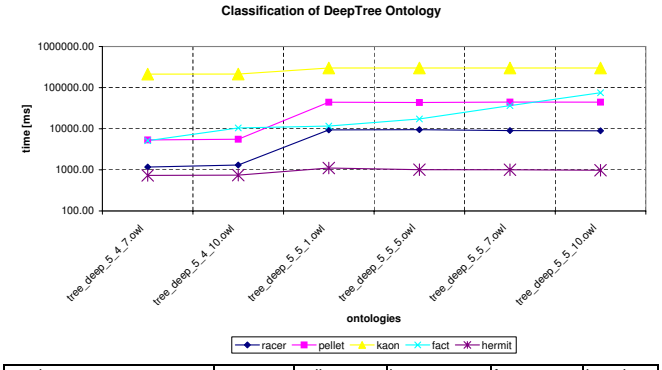


ontology	racer	pellet	kaon	fact	hermit
wine_0.owl	211.00	2075.00	3159.00	2914.00	368.00
wine_1.owl	218.00	1976.00	2674.00	25272.00	316.00
wine_2.owl	231.00	1811.00	3006.00	69356.00	441.00
wine_6.owl	234.00	1247.00	2976.00	300000.00	471.00
wine_7.owl	231.00	1446.00	3357.00	300000.00	381.00
wine_8.owl	237.00	300000.00	2665.00	300000.00	343.00
wine_9.owl	202.00	300000.00	2756.00	300000.00	329.00
wine_10.owl	259.00	300000.00	3163.00	300000.00	300000.00

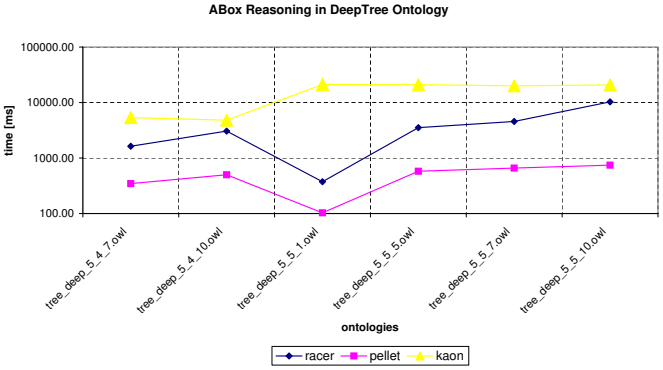


ontology	racer	pellet	kaon	fact	hermit
galen.owl	31465.00	300000.00	300000.00	2661.00	2475.00
galen-original.owl	300000.00	300000.00	300000.00	300000.00	8277.00
nci.owl	5355.00	14128.00	300000.00	1367.00	20716.00

Fig. 4. Execution time of the classification of the Wine ontology (Q_2) (left), and TBox benchmark ontologies (right).



ontology	racer	pellet	kaon	fact	hermit
tree_deep_5_4_7.owl	1169.00	5340.00	212810.00	5119.00	733.00
tree_deep_5_4_10.owl	1311.00	5561.00	213592.00	10462.00	743.00
tree_deep_5_5_1.owl	9380.00	44233.00	300000.00	11620.00	1107.00
tree_deep_5_5_5.owl	9492.00	43552.00	300000.00	17307.00	1008.00
tree_deep_5_5_7.owl	9031.00	44520.00	300000.00	36504.00	1003.00
tree_deep_5_5_10.owl	8902.00	44305.00	300000.00	75261.00	979.00



ontology	racer	pellet	kaon
tree_deep_5_4_7.owl	1634.00	346.00	5352.00
tree_deep_5_4_10.owl	3060.00	500.00	4808.00
tree_deep_5_5_1.owl	374.00	103.00	21106.00
tree_deep_5_5_5.owl	3532.00	579.00	20993.00
tree_deep_5_5_7.owl	4569.00	660.00	20013.00
tree_deep_5_5_10.owl	10307.00	746.00	20743.00

Fig. 5. Execution time of the classification (left) and ABox reasoning (right) in Deep-tree benchmark ontologies.

significantly affects the performance of both tableau and resolution based methods. We performed the following assertional query:

$$Q_1(x) = AmericanWine(x)$$

The results are shown in Fig. 4. We have also performed classification (Q_2) and measured load times for the ontology. Assertional queries followed the same overall pattern from previous tests with KAON2 and Pellet providing the most stable performance. Increased complexity of the TBox affects the performance of all the reasoners severely. Furthermore, Racer and HermiT show the best classification performance while there is a small gap between classification of the TBox between Pellet and KAON2. Fact and Pellet are severely influenced by the increased number of instances and were unable to compute the classification for ontologies with large number of individuals.

TBox ontologies. We have chosen GALEN, DOLCE and NCI as they are providing the most complex TBox, which can be handled by the existing reasoners. Unlike previous test, the only reasoner capable of computing classification of the GALEN original ontology was HermiT. Both Racer and Fact performed very well on DOLCE and NCI followed by Pellet. KAON2 was unable to compute classification for any of the given ontologies. It is clear that the performance of KAON2 lags behind the tableau based methods.

Deep-tree benchmark primarily consists of a set of ontologies with increasing TBox complexity. For every TBox, a set of ontologies with a growing number of ABox individuals is created [18]. Unlike previous ontologies Deep-tree is a purely synthetic benchmark without any relation to real-world ontologies. Fig.5 shows an overview of the performance for various reasoners. While Racer and Pellet performance seems solely dependent on the size of the ABox, KAON2 mainly depends on the complexity of the TBox and fails to compute classification beyond a certain point.

V. Related work

There is an extensive testing work on benchmarking OWL-DL ontologies such as TANCS [20] and the DL comparison suite [21]. Extensive set of benchmark suites and test results can be found in papers describing the actual reasoners such as [7], [9], [13], [11], [8].

Until recently relatively few real-world ontologies were available. Therefore number of synthetic benchmark tests such as Lehigh University Benchmark [22] were developed. A combination of synthetic and real-world ontologies was shown in [18] covering many aspects of the reasoners. Our paper extends such benchmarks and covers wider range of reasoners and ontologies, which target both ABox and TBox reasoning.

An automated test framework for ontological reasoning, which is based on real-world ontologies and also evaluates correctness of the answers was proposed in [23]. However, unlike our work it does not provide any ABox reasoning tests and relies solely on the DIG interface, which deteriorates the performance of Java-based reasoners. Another benchmarking of a number of reasoners against a broad range of realistic ontologies can be seen in [24]. The paper focuses only on the performance without any correctness checking of the output. A survey of existing real-world ontologies was described in [25]. Although it is not directly related to our paper, it provides an overview of the expressivity and other statistical data about the existing real-world ontologies.

VI. Conclusion

We have proposed a testing methodology and implemented a framework for automated testing of the OWL-DL reasoners. We have performed extensive tests of the performance on the ontologies from the Semantic Web community. While the performance of the tableau and hypertableau based methods was dominated in the TBox reasoning problems. The resolution based method as well as optimized tableau procedure implemented in Pellet dominates the conjunctive query answering and ABox reasoning problems. We have discussed the primary causes for this as a consequence of the particular optimizations. Currently, we are working on the evaluation of the proposed system on a wider scale of ontologies and implementation of the Web-based user interface.

Acknowledgment

The research reported in this paper has been partially financed by the EU/ICT within the projects FP7-213876, FP7-215024 and Slovak national projects, SEMCO-WS APVV-0391-06, APVV RPEU-0024-06, APVV RPEU-0029-06, VEGA 2/6103/6, VEGA 2/7098/27.

References

[1] B. T. Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, May 2001. [Online]. Available: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

[2] H. Zhuge, *The knowledge grid*. Singapore: World Scientific Publishing Co., 2004.

[3] ——, "China e-science knowledge grid environment," *IEEE Intelligent Systems*, vol. 19, no. 1, pp. 13–17, 2004.

[4] S. Bechhofer, M. Dean, F. V. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, G. Schreiber, and L. Stein, "Owl web ontology language reference, w3c proposed recommendation," http://www.w3.org/TR/owl-ref/, 2003, w3C Proposed Recommendation.

[5] G. Klyne and J. J. Carroll, "Resource description framework (rdf): Concepts and abstract syntax," Tech. Rep., 2004.

[6] H. Zhuge, P. Shi, Y. Xing, and C. He, "Transformation from OWL description to resource space model," in *The Semantic Web - ASWC 2006, First Asian Semantic Web Conference, Beijing, China, September 3-7, 2006, Proceedings*, ser. Lecture Notes in Computer Science, vol. 4185, 2006, pp. 4–23.

[7] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: a practical OWL-DL reasoner," *Journal of Web Semantics*, vol. 5, no. 2, pp. 51–53, 2007.

[8] V. Haarslev and R. Möller, "Description of the racer system and its applications," in *Proceedings International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August*, 2001, pp. 131–141.

[9] U. Hustadt, B. Motik, and U. Sattler, "Reasoning for Description Logics around SHIQ in a Resolution Framework," FZI, Germany, Tech. Rep. 3-8-04/04, 2004.

[10] D. Fensel and F. van Harmelen, "Unifying reasoning and search to web scale," *IEEE Internet Computing*, vol. 11, no. 2, pp. 96–95, 2007.

[11] B. Motik, R. Shearer, and I. Horrocks, "Optimized Reasoning in Description Logics using Hypertableaux," in *Proc. of the 21st Conference on Automated Deduction (CADE-21)*, ser. LNAI, F. Pfenning, Ed., vol. 4603. Bremen, Germany: Springer, July 17–20 2007, pp. 67–83.

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

[13] I. Horrocks, "The FaCT system," *In Automated Reasoning with Analytic Tableaux and Related Methods: International Conference (Tableaux'98), volume 1397*, vol. 1397, pp. 307–350, 1998.

[14] P. Patel-Schneider, "DLP system description," *Automated Deduction - CADE-17*, vol. 1831/2000, pp. 297–301, 2000.

[15] R. M. MacGregor, "Inside the LOOM description classifier," *SIGART Bull.*, vol. 2, no. 3, pp. 88–92, 1991.

[16] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick, "CLASSIC: a structural data model for objects," in *SIGMOD '89: Proceedings of the 1989 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 1989, pp. 58–67.

[17] A. Kiryakov, D. Ognyanov, and D. Manov, "OWLIM a pragmatic semantic repository for owl," in *Web Information Systems Engineering WISE 2005 Workshops*. IEEE Computer Society, 2005, pp. 182–192.

[18] T. Weithöner, T. Liebig, M. Luther, S. Böhm, F. W. von Henke, and O. Noppens, "Real-world reasoning with owl." in *ESWC*, ser. Lecture Notes in Computer Science, E. Franconi, M. Kifer, and W. May, Eds., vol. 4519. Springer, 2007, pp. 296–310.

[19] K-Wf Grid, "K-Wf Grid technical annex," http://www.kwfgrid.net, 2004.

[20] F. Massacci and F. M. Donini, "Design and results of tancs-2000 non-classical (modal) systems comparison," in *TABLEAUX '00: Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*. London, UK: Springer-Verlag, 2000, pp. 52–56.

[21] I. Horrocks and P. F. Patel-Schneider, "Dl systems comparison (summary relation)," in *Description Logics*, ser. CEUR Workshop Proceedings, E. Franconi, G. D. Giacomo, R. M. MacGregor, W. Nutt, and C. A. Welty, Eds., vol. 11. CEUR-WS.org, 1998.

[22] Y. Guo, Z. Pan, and J. Heflin, "Lubm: A benchmark for owl knowledge base systems." *J. Web Sem.*, vol. 3, no. 2-3, pp. 158–182, 2005.

[23] I. H. Tom Gardiner, Dmitry Tsarkov, "Framework for an automated comparison of description logic reasoners," in *Proceedings of the International Semantic Web Conference - ISWC 2006*. Springer Berlin, 2006, pp. 654–667.

[24] Z. Pan, "Benchmarking dl reasoners using realistic ontologies," in *Proceedings of the OWL: Experiences and Directions Workshop. Galway*, 2005. [Online]. Available: http://www.mindswap.org/2005/OWLWorkshop/sub6.pdf

[25] T. D. Wang, B. Parsia, and J. A. Hendler, "A survey of the web ontology landscape." in *International Semantic Web Conference*, 2006, pp. 682–694.