Benchmarking DL Reasoners Using Realistic Ontologies

Zhengxiang Pan

Bell Labs Research and Lehigh University
zhp2@lehigh.edu

Abstract. We did a preliminary benchmark on DL reasoners using real world OWL ontologies. First hand experiences on OWL ontologies and reasoning services available to OWL are described and discussed.

1 Introduction

We developed a benchmark to evaluate the performances of state-of-the-art expressive Description Logic reasoners using an empirical approach. The outcomes are expected to illustrate the effectiveness of DL reasoners that vary in optimization techniques and implementation strategies. Some observations on the deployment of OWL language are also made and discussed.

We conducted the benchmark toward three well-known DL reasoners: Racer, FaCT++ and Pellet. The benchmark data consists of more than one hundred most deployed OWL ontologies across many domains, from life science [1] to geographic [2], from food and wines [3] to stock exchange [4].

Because of the heterogeneous interfaces (DIG and HTTP etc.) being used in the different systems, this benchmark should be regarded as more qualitative than quantitative, i.e. precise time counting is of little interests here. Instead, we will try to analyze the detailed outputs and logs of those reasoning systems and extract useful informations.

2 Background

Due to the correspondence between OWL and description logics, reasoning support to OWL applications heavily relies on description logic reasoners. Benchmarks and evaluations on DL reasoners thus become an important task that concerns with OWL.

A number of efforts have been made on benchmarking DL reasoner. Some of them take the road of generating synthetic formulas randomly [5] [6] [7]. Although this approach could lead to a more comprehensive benchmark on reasoners, the parameters of those generators need to be fine tuned by sampling the realistic KBs. In contrast, using the real world KBs as test data would be more feasible and efficient, the results would be more easily interpreted and used by end users. Part of the test in [8] was using realistic TBoxes, but only 8 KBs were included at that time. An analysis of a DAML+OIL ontology collection in [9] characterized and categorized the real-world ontologies but the depicted benchmark was not implemented and conducted.

3 Experiments

3.1 Target Systems

Three DL reasoners were chosen to run the benchmark: Racer, Pellet and FaCT++. Certainly there are other DL reasoners worth benchmarking but were not tested due to various constraints.

According to [10], Racer implementation employed tableaux calculus for $\mathcal{SHIQ}$ as well as the following optimization techniques: dependency-directed backtracking and DPLL-style semantic branching, transformation of axioms (GCIs), model caching and model merging.

Pellet is claimed to be sound and complete on $\mathcal{SHIN}(\mathcal{D})$ and $\mathcal{SHON}(\mathcal{D})$ [11]. It implements TBox partitioning, absorption and lazy unfolding plus dependency directed backjumping, semantic branching and early blocking strategies. It also supports datatype reasoning and uses some optimizations in Abox query answering.

Not quite similar to FaCT, FaCT++ implemented a new tableaux decision procedure for $\mathcal{SHOIQ}$ [12]. Being under its early stage of development, FaCT++ has very limited user interface and no API is available.

3.2 Test Data

Our test data consists of 135 real world OWL ontologies. They were submitted by the ontology authors and users from different domains.

Our original plan was to use an OWL-aware crawler to crawl OWL ontologies available on the web. However, we did not find such a tool. Hence we turned to finding a large collection of OWL ontologies. Among handful candidates, Schemaweb [13] which is a RDF schemas directory became our choice. A java program was developed then to read and identify those schemas.

We fed 250 urls indexed on Schemaweb into our program and we identified 135 OWL ontologies among them. Totally 5897 classes and 2601 relations were recorded out of these OWL ontologies.

Having their subjects distributed in a broad range of domains, these OWL ontologies also vary in size, constructs being used and complexity. Thus, we argue these 135 OWL ontologies largely represent the current usage of OWL language in practical despite that they are just a small portion of existing OWL documents.

3.3 Experiment Configurations

The experiments were done on a Linux box featuring an Intel(R) Pentium(R) 4 CPU at 2.6GHz and 1 giga bytes main memory.

For each target system, script or special handling program were written to direct the benchmark. No matter how it was being executed, the benchmark is the iteration of the following steps:

1. Clear the memory and cache in the application.
2. Read in the next ontology in the test set.
3. Do classification.

A time limit of one hour (3600 seconds) was set for each ontology, meaning that any processes regarding a particular ontology will be aborted if aggregated CPU time exceeds 3600 seconds.
Racer:
RacerPro 1.8.1 has recently released as a commercial software; however in this paper we used the last free-for-research version Racer 1.7. A racer instruction file was created to run the benchmark. Each ontology corresponds to four commands in that file. First two commands (DELETE-ALL-ABOXES) and (DELETE-ALL-TBOXES) cleaned up the memory. Then (OWL-READ-DOCUMENT "*url*") command asked Racer to read in the specified ontology. At the end, (TBOX-COHERENT-P) and (ABOX-CONSISTENT-P) invoked the classifications in the Reasoner.
Pellet:
We used the Pellet 1.1.0 released on 05/07/2004. A script file was created to manage the benchmark. Each parameterized execution of Pellet would read one ontology and do the classification. Since each ontology was processed by a fresh start of the Pellet, no need to clean the memory and cache in this case.
FaCT++:
As part of the aforementioned limitations, FaCT++ doesn't take OWL documents directly nor any remote files. A utility program digFaCT++ takes local files in DIG [14]. In order to make the benchmark working, we developed a java program to translate the OWL ontologies into DIG format and store them locally. In the benchmark script, each execution of digFaCT++ was supplied two parameters. One is a tell-document in DIG corresponding to one ontology, the other is a simple query file that only ask if TOP is satisfiable. This simple query was used here to invoke the classification in FaCT++.

4 Results

Racer finished the benchmark in about 15 minutes. It successfully made TBox classifications on 108 ontologies, 101 of which were found to be consistent. It also made successful consistency check on ABox for 92 ontologies, 83 of which were found to be consistent. For those aborted tasks and inconsistent ontologies, Racer reported 117 errors, about one third of which is due to the syntax errors or usages beyond the scope of OWL DL.

Pellet had done classifications on 103 ontologies within the time limit. It spent almost 2 hours (6814 seconds) on these ontologies. Interestingly, all these finished ontologies were classified to be consistent. However, Pellet automatically did somethings more than just classification. It validates the species of the ontologies and tries to repair OWL Full ontologies if they are missing type triples [11]. In our benchmark result, 70 out of 103 ontologies were validated as OWL Full, 23 and 10 for DL and Lite respectively.

Except timed out for 3 ontologies, FaCT++ had done the remaining in nearly 30 seconds. Its log recorded that it only spent 2.6 seconds on classifications of the 121 ontologies, which were all successfully classified. Note the time of parsing and I/O was not included, nor was the time spent on translating OWL into DIG. Nevertheless, this kind of performance was very impressive.

System	Consistent	Inconsistent	Timed out	Aborted
Racer(Tbox)	101	7	0	27
Pellet	103	0	17	15
Fact++	121	0	3	11

Table 1. The Results of Classification: Performed on 135 ontologies

5 Discussion

Here we summarize some interesting observations from our benchmark results. They could potentially give us some helpful hints on ontology authoring as well as the design and implementation of reasoners.

Firstly, the test data and the output of reasoners gave us a good chance to characterize the current usage of OWL language. Based on the result from Pellet, more than 70% of the classified ontologies are OWL Full, more than three quarters of these OWL Full ones can be validated as OWL DL just by adding some statements, like type triples. Racer also found out 22 cases where transitive properties were used in cardinality restrictions, legitimate only in OWL Full.

In addition, we used the WonderWeb validator [15] to validate the species of the rest ontologies. By adding these up we get figure 1, showing that the majority of the test KBs are OWL Full. Note the "unknown" category was for those ontologies that caused errors on the validator.

We assume that only few authors intended to create an ontology in OWL Full, because of the extreme difficulties in finding reasoning support. Thus, an ontology editor with built-in validator and heuristic non-DL finder is highly desired and should be widely adopted. Some efforts has been made toward this direction, such as [16].

Secondly, the performances of the reasoners varied a lot. Although they are not directly comparable due to the different input formats (OWL v.s. DIG) and side-functionalities (species checking etc.), the results implies the effectiveness of some optimization techniques being deployed. Apparently, FaCT++ completed the most testing ontologies using the least time. Within the scope of classification, FaCT++ significantly pushed the baseline of DL reasoners to a new high.

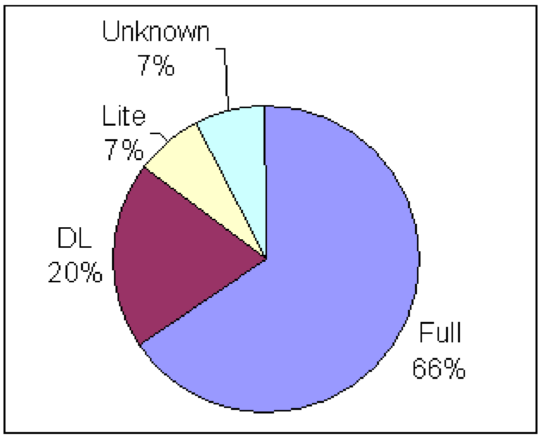

Fig. 1. Percentages of Each Species of OWL in Test Data

However, FaCT++ provides very limited services compare to other rivals. So far it only supports queries through DIG, which is not very expressive in ABox retrieving.

Pellet, on the other hand, done least TBox classifications with most time. Unfortunately, we have no way to figure out how much time was actually spent on those extra functionalities such as species checking. Figure 2 shows that classification time in Pellet increased nearly exponentially, no wonder 17 testing ontologies were timed out. Interestingly, we found out that some of the ontologies that Pellet spent a huge amount of time on (but finished) were the ones timed out or failed by FaCT++. This suggests that to some extent, Pellet is more resilient to non-trivial ontologies.

For most of the testing documents, Racer was not as fast as FaCT++, but it never timed out. This intrigues a dilemma on the implementation strategies: give time or give up? Guaranteed termination is a nice property but sometimes resilience is also desired. One possible solution is to allow users to customize the time-out settings for each execution.

Above all, the performance of a DL reasoner is affected by the following factors:

– The quality of inputs. DL reasoners are not intended to perform on non-DL ontologies. Reasoner should be able to identify its capability on the given knowledge base before long deliberations.
– The optimization techniques. Other experiences [17] show that cyclic axioms, inverse roles and nominals are "killer" constructs for DL reasoners. New optimizations should target these cases.

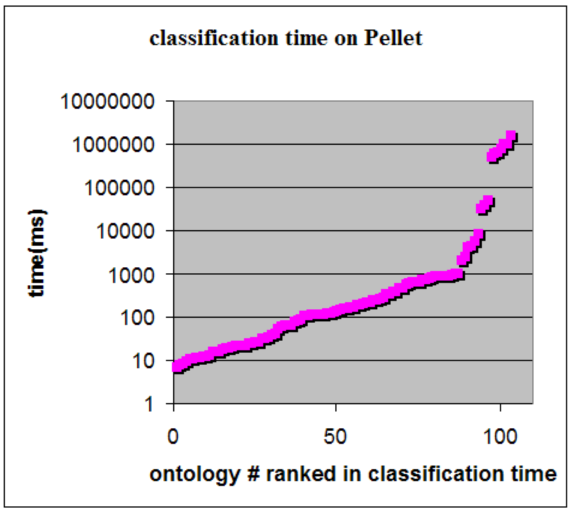

Fig. 2. Performance degradation of Pellet

– The feasibility for customization. Different applications have different constraints or preferences on speed and/or expressiveness. They require different side-functionalities even just for classification.

6 Conclusion

We performed a preliminary benchmark on three state-of-the-art DL reasoners: Racer, Pellet and FaCT++. They vary from each other in many aspects, even in programming languages: Lisp, Java and C++ respectively. Real world OWL ontologies across various domains were used as test KBs. Observations on the characteristics of those OWL ontologies as well as the performances of the reasoners were reported and discussed. We do not intend to use the results as direct reflection of those systems' overall performances, for that this simple benchmark is not systematic enough and only focus on TBox classification at this time.

There are a couple of future directions on this work. Firstly we should identify and study case by case on these non-trivial KBs, i.e. the ones timed out, failed or spent a lot of time. These will intrigue research directions on DL reasoner optimizations. Secondly we need to formalize our benchmark by making the targeting reasoners more comparable, probably wrapping them in the same API. Furthermore, the detailed relationship between optimization techniques and performances should be analyzed.

7 Acknowledgement

This work was carried out while the author was visiting Bell Labs at Murray Hill, New Jersey, USA.

References

1. (http://www.cs.man.ac.uk/ wroec/mygrid/ontology/mygrid.owl)
2. (http://loki.cae.drexel.edu/ wbs/ontology/2003/10/iso-metadata.owl)
3. (http://www.w3.org/TR/owl-guide/wine.rdf)
4. (http://www.daml.org/2001/10/html/nyse-ont)
5. Q. Elhail, M.R., Ycart, B.: Generating random benchmarks for description logics. In: Proc. of DL'98. Volume 11. (1998)
6. Patel-Schneider, P.F., Sebastiani, R.: A new general method to generate random modal formulae for testing decision procedures. J. Artif. Intell. Res. (JAIR) **18** (2003) 351–389
7. Hladik, J.: A generator for description logic formulas. In Horrocks, I., Sattler, U., Wolter, F., eds.: Proceedings of DL 2005, CEUR-WS (2005) Available from `ceur-ws.org`.
8. Horrocks, I., Patel-Schneider, P.F.: Dl systems comparison (summary relation). In: Description Logics. (1998)
9. Tempich, C., Volz, R.: Towards a benchmark for semantic web reasoners - an analysis of the daml ontology library. In: EON. (2003)
10. Haarslev, V., Möller, R.: Racer system description. In Goré, R., Leitsch, A., Nipkow, T., eds.: International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, Siena, Italy, Springer-Verlag (2001) 701–705
11. Parsia, B., Sirin, E.: Pellet: An owl dl reasoner. In: Proc. International Semantic Web Conference. (2004)
12. Horrocks, I., Sattler, U.: A tableaux decision procedure for $\mathcal{SHOIQ}$. In: Proceedings of Nineteenth International Joint Conference on Artificial Intelligence. (2005)
13. (http://www.schemaweb.info)
14. Bechhofer, S.: The DIG Descriprion Logic Interface: DIG/1.1. (2003)
15. (http://phoebus.cs.man.ac.uk:9999/OWL/Validator)
16. Bechhofer, S., Volz, R.: Patching syntax in owl ontologies. In: Proceedings of the Third International Semantic Web Conference. (2003)
17. Haarslev, V., Möller, R., Wessel, M.: Description logic inference technology: Lessions learned in the trenches. In Horrocks, I., Sattler, U., Wolter, F., eds.: Proc. International Workshop on Description Logics. (2005)