Automated Benchmarking of Description Logic Reasoners

Tom Gardiner, Ian Horrocks, Dmitry Tsarkov
University of Manchester
Manchester, UK
{gardiner|horrocks|tsarkov}@cs.man.ac.uk

May 12, 2006

1 Introduction

Reasoning for expressive DLs implemented in state-of-the-art systems has high worst case complexity. The hope/claim is, however, that these systems perform well in "realistic" applications. In practice, this means in ontology applications. To check the validity of this claim it is necessary to test the performance of these systems with (the widest possible range of) ontologies derived from applications.

In addition, testing is useful in order to check the correctness of implementations. In small examples, it may be easy to check the correctness of a system's reasoning. However, for typical real-world examples, manual checking is not feasible. In these instances, the best (perhaps the only) way to check correctness is often by checking for consistency with the reasoning of other existing systems.

Real-world ontologies vary considerably in their size and expressivity. While they are all valuable test cases, it is still important to understand each ontology's properties in order to provide efficient and relevant testing.

System developers find this particularly useful, as it helps them to identify weaknesses in their systems and to devise and test new optimisations. Finally, testing is also useful for the developers and users of applications as they can use benchmarking results to determine if (the performance of) a DL reasoner is likely to satisfy their requirements, and if so which reasoner is likely to perform best in their application.

2 Background and Related Work

For the above mentioned reasons, there is extensive existing work on benchmarking DL (as well as modal logic) reasoners. E.g., TANCS comparisons and benchmark suites [10], DL comparisons and benchmark suite [1], work on M-SPASS [8], work on FaCT and DLP [7, 6], the OWL benchmark suite and test results, and various test results from papers describing systems such as FaCT++ [13], Pellet (`http://www.mindswap.org/2003/pellet/`), Racer [5], KAON2 (`http://kaon2.semanticweb.org/`), Vampire [12], etc.

Due to the fact that relatively few (large and/or interesting) ontologies were available, earlier tests often used artificially generated test data. The Lehigh University Benchmark [4], for example, used a synthetic ontology and randomly generated data to test the capabilities of knowledge base systems using specific weightings to compare systems on characteristics of interest. Results from such tests are, however, of doubtful relevance when gauging performance on ontologies. The popularity of OWL has meant that many more ontologies are now available, and recent benchmarking work has focused on testing performance with such ontologies.

One such example [11] involved benchmarking a number of reasoners against a broad range of realistic ontologies. However, not all reasoners used in that comparison supports OWL as an input language, so quantitative comparison of performance would have been difficult/un-justified. From the other hand, the DIG interface [2] is recognised as a preferred choice by application developers and thus is implemented into a wide range of DL Reasoners.

Our work builds on these earlier efforts, taking advantage of the DIG standard to provide a generic benchmarking suite that allows the automatic quantitative testing and comparison of DL Reasoners on real-world examples with relevant properties. We aim to make the testing process as autonomous as possible, taking care, for example, of (re)starting and stopping reasoners as necessary, and the analysis of results be as flexible as possible, by allowing for arbitrary SQL queries against the collected data. We also aim to provide, as a publicly available resource, a library of test ontologies where each ontology has been checked for expressivity and syntactic conformance, translated into DIG syntax (which is much easier to work with that OWL's RDF/XML syntax), and includes (where possible) results (such as the concept hierarchy) that can be used for testing the correctness of reasoning systems.

3 Methodology

The system has two main functions. The first is to process ontologies and add them to the library, and the second is to benchmark one or more reasoners using the ontology library.

When processing ontologies, the system takes as input a list of OWL-ontology URI's. Before they can be used in testing, some preprocessing of these ontologies is required. The process involves generating valuable meta-data about each ontology, as well as converting each of the OWL-ontologies to DIG.

The meta-data is generated by code written for SWOOP [9], and provides the details of the expressivity (i.e. the constructs present in the ontology) together with the number of classes, object properties, data properties, individuals, class axioms, property axioms and individual axioms present. This is invaluable information in helping to understand the meaning of any results obtained through testing, in finding, for example, strengths and weaknesses of particular systems. The OWL-to-DIG conversion uses the OWL-API (`http://sourceforge.net/projects/owlapi`). This process is far from trivial as OWL's RDF syntax is extremely complex, and it is easy to (inadvertently) cause ontologies to be outside of OWL DL, e.g., by simply forgetting to explicitly type every object. Moreover, the DIG interface supports only the most basic of data types, such as Strings and Integers. The result is that many of the available OWL Ontologies we found could not be successfully converted to DIG.

Local copies are stored of both the OWL Ontology and the DIG version. This is not only for efficiency during the testing, but also to ensure consistency (as online ontologies rarely remain static). Moreover, this allows us to fix trivial errors in the OWL ontologies so that they can be used for testing purposes. The locations of these files, together with their properties/meta-data, are stored as database entries for easy access and manipulation.

The main function of the benchmark suite itself is timing the classification of each ontology by each Reasoner. To promote fairness, each Reasoner is terminated and then restarted for every ontology.

A problem with trying to compare different Reasoners is that they may perform tasks in different ways. For example, they may vary in the way in which they perform each part of the reasoning: some may take an "eager" approach, fully classifying the whole ontology and caching the results as soon as it is received; others may take a "lazy" approach, only performing reasoning tasks as required in order to answer queries. To try to get around this problem, we use a five step test, for each ontology, that forces every reasoners to fully classify that ontology. The steps are as follows:

1. TELL the reasoner the full ontology
2. ASK for all the concepts in the ontology
3. ASK for the satisfiability of the TOP concept
4. ASK for the satisfiability of all the concepts in the ontology
5. ASK for the ontology taxonomy (parents and children of all concepts)

Each of these individual steps are timed, providing interesting information about when different reasoners do most their work. It is, however, the total time for this complete (classification) test that we are most interested in.

Each test will end in one of three ways. It will either complete successfully, fail due to lack of time or fail for some other reasons. The latter may include failure due to lack of run-time memory, failure because the reasoner could not parse the ontology successfully, etc.

The benchmark suite is fully automatic, dealing with most errors autonomously, meaning that the testing can be left to run over-night or over a week-end (which may be necessary when using a large time-out). All data is recorded in a MySQL database, making it easy for the user to view and analyse the data in a variety of ways.

As discussed in Section 1, in order to get a clearer indication of how DL Reasoners perform in the real world, we aim to build a large library of OWL ontologies from those that are publicly available. Currently, our library contains a little over 300 OWL-RDF Ontologies, but only 172 of these could successfully be converted to DIG. This has, however, provided us with a total of just under 72,000 classes and over 30,000 individuals in a DIG format. Only 18% of the ontologies were at least $\mathcal{ALC}$, which suggests that the majority of real-world ontologies aren't in fact very complex, but it also means we have a comfortable number of "interesting" examples too.

4 Testing

Our system is currently fully automatic and runs the classification tests successfully through our whole library. It does not, however, at this stage verify the correctness of each Reasoner's answers to the queries (from steps 2-5) and how they compare to the answers given by other Reasoners. This means that our measure of success is, for now, merely an indication that the Reasoner received and parsed the DIG successfully and returned a valid DIG response. This is generally a good indication, but should only be considered a preliminary result.

We have performed some tests on our system, as it stands, and we provide here some examples of the kinds of information that our system can produce.

FaCT++ v1.1.3, KAON2, Pellet v1.3 and RacerPro v1.8.1 are four of the most widely used OWL/DIG reasoners, and we therefore decided to use these to test the current capabilities of our system. The tests were performed using an Intel Pentium-M Processor 1.60 GHz and 1Gb of main memory on Windows XP. The time-out period was set to 10 minutes (in real time). Pellet and KAON2 are java applications, and for these tests were run with a maximum heap space of 200Mb. RacerPro and FaCT++ were left to run on their default settings. Our system does not try to optimise the performance of the Reasoners for particular ontologies, as we believe this is the job of the Reasoners themselves, not the application user.

Table 1 shows how the individual Reasoners performed firstly on all our ontologies and then on Ontologies which have particular characteristics. Finally, it shows their performance on OWL-Lite ontologies, which includes all those with expressivity up to SHIF.

In order to determine which were the most "challenging" ontologies (w.r.t. reasoning), we tried to order ontologies according to the difficulty of reasoning with them. To do this, we used all the ontologies that were successfully classified by at least two Reasoners and then ordered these by their average classification

Type	Status	FaCT++	KAON2	Pellet	RacerPro
All	Success	138	45	152	110
All	Failed	29	124	18	62
All	TimedOut	5	3	2	0
Nominals	Success	7	3	9	7
Nominals	Failed	4	9	3	5
Nominals	TimedOut	1	0	0	0
TransRoles	Success	15	9	18	13
TransRoles	Failed	4	11	3	9
TransRoles	TimedOut	3	2	1	0
Datatypes	Success	102	12	114	75
Datatypes	Failed	23	15	13	52
Datatypes	TimedOut	2	0	0	0
OWL-Lite	Success	33	31	33	34
OWL-Lite	Failed	5	6	5	6
OWL-Lite	TimedOut	2	3	2	0

Table 1: Sample of Overall Performance

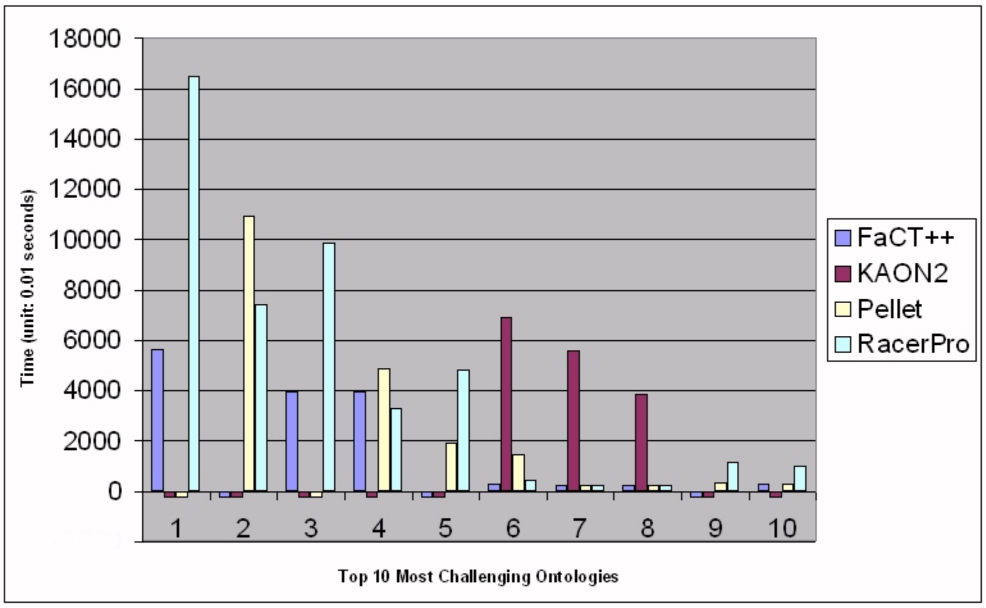

Figure 1: Comparison of Reasoners on the Top 10 Most Challenging Ontologies

Ontology	Expressivity	nClass	nIndiv	URL
1	DL-Lite	27652	0	http://...logy/nciOncology.owl
2	$\mathcal{SHF}$	3097	0	http://...ibrary/not-galen.owl
3	$\mathcal{ALR}+$	20526	0	http://archive.godatabase.org/
4	$\mathcal{SHF}$	2749	0	http://...Ontologies/galen.owl
5	RDFS(DL)	1108	3635	http://...world-fact-book.daml
6	RDFS(DL)	1514	0	http://...logy/data/center.owl
7	$\mathcal{ALCF}$(D)	87	0	http://...a/pizza/20041007.owl
8	$\mathcal{SHIF}$	37	0	http://...s/DOLCE-Lite/397.owl
9	RDFS(DL)	4	1899	http://...nt/AirportCodes.daml
10	$\mathcal{ALR}+\mathcal{HI}$(D)	5	2744	http://...ogicUnits/2003/09/hu

Table 2: Properties of Top 10 Most Time-consuming Ontologies

Reasoner	Tells	ConceptList	SatOfTop	SatOfClasses	Hierarchy
FaCT++	23%	35%	11%	9%	21%
KAON2	47%	45%	0%	3%	5%
Pellet	70%	20%	1%	2%	6%
RacerPro	58%	11%	4%	9%	19%

Table 3: Average Division of Task Time

time. Figure 1 shows the amount of time each Reasoner took to classify the 10 most challenging ontologies according to this measure (where negative time represents a failure to classify). Table 2 then shows some of the interesting information that is available on these "Top 10" Ontologies.

This table is useful in helping us understand what makes these particular Ontologies so time-consuming to reason over. In the case of the NCI and Gene Ontology's (1st and 3rd), it can be clearly seen that it is their shear size that provides the challenge. The 5th, 9th and 10th (world-fact-book, AirportCodes and Hydrolic Units) make up for their number of classes with an extensive array of individuals. Whereas Galen (2nd and 4th) simply uses some very complicated constructs and deep role hierarchy.

Our final table, Table 3, shows the average proportion of each classification test that each Reasoner spent on the separate tasks. This shows, for example, that Pellet performs a lot of caching on receiving the Ontology (TELLS), while FaCT++ does relatively little until the first ASK query.

5 Discussion

As we mentioned in the introduction, testing is useful for reasoner and tool developers as well as for users. Building on existing work, we have developed a system for testing reasoners with available ontologies. The benefits of our approach include autonomous testing, flexible analysis of results and the development of a test library that should be a valuable resource for both the DL and ontology community. We will continue to extend the library, and will add classification results from tested reasoners so that correctness testing can also be performed.

While there are an increasingly large array of OWL-Ontologies available for public use, other Ontology formats (e.g. OBO: the Open Biomedical Ontologies, `http://obo.sourceforge.net`) are still widely in use and would make for valuable test examples. It is also the case, as describe in [3], that a large proportion of the available OWL-Full Ontologies, could in fact be validated as OWL-DL, just by adding a few extra clarifying statements. This means that of the 162 Ontologies that we had to throw away, many could be useful examples with a little work. In the future we hope to use these observations, together with any external contributions, to considerably increase the size of our ontology library.

The results produced by our tests provide an interesting insight into the variety and depth of information that can be extracted from such testing/benchmarking. However, for the system and its results to become a valuable resource, we need to test their correctness. We are currently assuming that both the OWL-to-DIG conversions and the Reasoner's responses are all valid and correct.

With regard to the OWL-API's conversions, this was the utility built alongside the original DIG specification. We therefore argue that this is the best conversion available and that our assumption is justified.

Regarding the responses, as discussed earlier, they can be almost impossible to check for correctness. Our best option is therefore to analyse the difference in responses received from different reasoners, and this route is thus one we aim to explore further. It will be interesting to see if reasoners (that should, in theory, all produce the same inferences to the same problems) will actually agree on the test ontologies.

So far we have focused on testing Tbox reasoning (classification). Although the use of nominals in $\mathcal{SHOIQ}$ blurs the separation between Tbox and Abox, it would still be useful to explicitly test Abox reasoning, e.g., by asking for the instances of some query class. This functionality will be added in a future version of the system.

Apart from the future work described above, there are a number of extensions to our benchmarking system that would enhance its utility. Allowing users to define their own customised test, rather than the 5 step classification we are using, is one example that would allow Reasoner developers to test specific optimisations and implementations as they are developed. Other relevant tests would include testing how multiple concurrent tests on a Reasoner affects performance, as well as simply not restarting a Reasoner between tests.

We intend for the whole system, including the ontology library, to be available for open-source use in the near future.

References

[1] P. Balsiger and A. Heuerding. Comparison of theorem provers for modal logics. In *Proceedings of Tableaux'98*, May 1998.

[2] Sean Bechhofer, Ralf Möller, and Peter Crowther. The DIG description logic interface. In *Proceedings of DL2003 International Workshop on Description Logics*, September 2003.

[3] Sean Bechhofer and Raphael Volz. Patching syntax in OWL ontologies. In *Proceedings of 3rd International Semantic Web Conference, ISWC*, 2004.

[4] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2):158–182, 2005.

[5] V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, Siena, Italy*, pages 701–705. Springer-Verlag, 2001.

[6] I. Horrocks. Benchmark analysis with FaCT. In *Proc. of TABLEAUX 2000*, number 1847, pages 62–66. Springer-Verlag, 2000.

[7] I. Horrocks and P. F. Patel-Schneider. FaCT and DLP. In *Proc. of TABLEAUX 98*, pages 27–30, 1998.

[8] U. Hustadt and R. A. Schmidt. MSPASS: Modal reasoning by translation and first-order resolution. 2000.

[9] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, and James Hendler. Swoop: A 'web' ontology editing browser. *Journal of Web Semantics*, 4(2), 2005.

[10] Fabio Massacci and Francesco M. Donini. Design and results of TANCS-00. volume 1847, 2000.

[11] Zhengxiang Pan. Benchmarking DL reasoners using realistic ontologies. In *Proc. of the OWL: Experiences and Directions Workshop*, 2005.

[12] A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.

[13] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, 2006. To Appear.