OWL2Bench: Towards a Customizable Benchmark for OWL 2 Reasoners

Gunjan Singh[1], Ashwat Kumar[1], Kanav Bhagat[1], Sumit Bhatia[2], and Raghava Mutharaju[1]

[1] Knowledgeable Computing and Reasoning Lab, IIIT-Delhi, India
{gunjans, ashwat16023, kanav16046, raghava.mutharaju}@iiitd.ac.in
[2] IBM Research AI, New Delhi, India
sumitbhatia@in.ibm.com

1 Introduction

In the past decade, there has been remarkable progress towards the development of reasoners[3] that involve expressive ontology languages such as OWL 2 [3]. However, they still do not scale well on expressive language profiles (OWL 2 DL). To build better quality reasoners, developers need to find and improve the performance bottlenecks of their existing systems [11]. A reasoner benchmark aids the reasoner developers to evaluate their system's performance and deal with the limitations. Furthermore, it paves the way for further research to improve performance and functionality. In particular, a reasoner needs to be evaluated from several aspects such as support for different language constructs and their combinations, their effect on reasoning performance, ability to handle large ontologies, and capability to handle queries that involve reasoning. Although there are some existing ontology benchmarks, they are limited in scope. LUBM [2] and UOBM [7] are based on the older version of OWL (OWL 1). OntoBench [6] supports OWL 2 profiles but does not evaluate reasoner performance. ORE benchmark framework[4] does not consider evaluation in the context of varying sizes of an ontology. In essence, no existing benchmark covers all the above-mentioned aspects for reasoner evaluation. Here, we describe our ongoing efforts towards building a customizable ontology benchmark for OWL 2 reasoners named OWL2Bench[5] (to be presented at the ISWC 2020 Resources Track) [8]. We also briefly discuss the planned future extensions to the benchmark.

2 OWL2Bench

OWL2Bench is an extension of the well known University Ontology Benchmark (UOBM) [7]. It consists of three major components: a fixed TBox for each OWL

[3] http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/
[4] https://github.com/ykazakov/ore-2015-competition-framework
[5] https://github.com/kracr/owl2bench

2 profile (EL, QL, RL, and DL), an ABox generator that can generate ABox of varying sizes for the corresponding TBox, and a fixed set of 22 SPARQL queries that involve reasoning. Thus, it allows users to benchmark three aspects of the reasoners - support for different OWL 2 profiles, scalability in terms of ABox size, and query performance. Moreover, the set of SPARQL queries also enables benchmarking of SPARQL query engines that support OWL 2 reasoning. The TBox for each profile was created by enriching UOBM's university ontology with the supported constructs. In order to generate varying size ABox, two user inputs are required, the number of universities and the OWL 2 profile (EL, QL, RL, or DL) of interest. The generated instance data complies with the schema defined in the TBox of the selected profile, and the size depends on the number of universities. For one university, by default, approximately 50,000 ABox axioms are generated.

To demonstrate the utility of OWL2Bench, we ran our benchmark on six reasoners, ELK [5], HermiT [1], JFact[6], Konclude [10], Openllet[7], and Pellet [9] for three reasoning tasks, i.e., consistency checking, classification, and realisation. We also evaluated two SPARQL query engines, Stardog[8] and GraphDB[9], on SPARQL queries in terms of their loading time and query response time. During our evaluation, we identified possible issues with these systems (some of which have already been communicated with the developers) that need to be fixed and could pave the way for further research in the development of reasoners and query engines. The performance of the reasoners on OWL2Bench is shown in Figure 1.

For our experiments, we set the heap space to 24 GB and time-out to 90 minutes. Most of the reasoners timed-out for even a small number of universities (except for QL profile). Although Konclude is much faster, it requires a lot of memory and could not perform any reasoning task after 50 universities. For the EL profile, both Konclude and ELK performed exceptionally well in terms of time taken, but ELK is better due to its low memory requirements. For the RL profile, most reasoners timed-out on larger ontologies. In the case of OWL 2 DL, Konclude, HermiT, and Pellet were able to complete the consistency checking task only (for 1, 2, and 5 universities, respectively). However, we observed some inconsistency in the results of Pellet. Other evaluations were time-outs. More details about the benchmark and the results are available in the full-version of our paper [8].

3 Proposed Extensions

When selecting a reasoner to use, there could be many considerations. One of the approaches is to select the best possible reasoner in terms of its efficiency and scalability. OWL2Bench can be used for this requirement. The other approach is

[6] http://jfact.sourceforge.net/
[7] https://github.com/Galigator/openllet
[8] https://www.stardog.com/
[9] http://graphdb.ontotext.com/

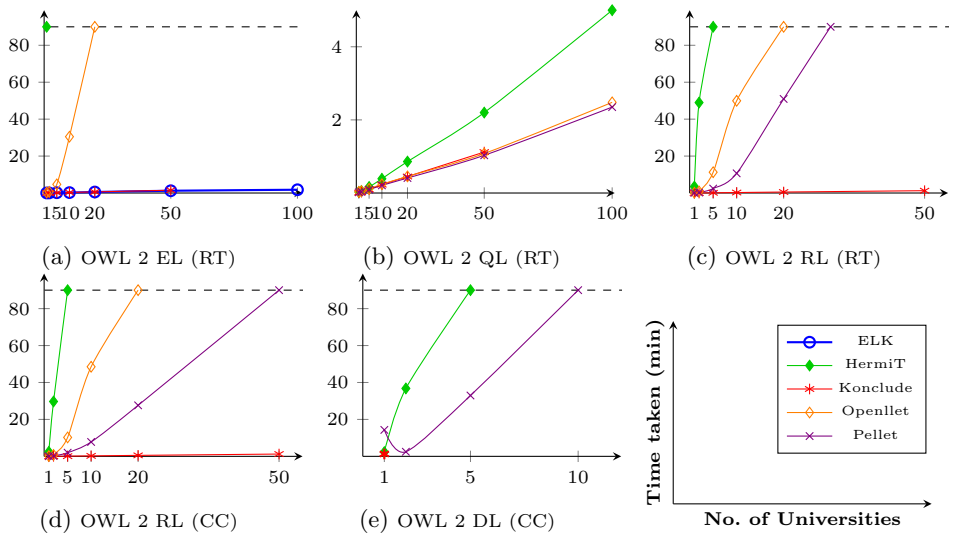

Fig. 1: Time taken in minutes (Y-axis) by reasoners on ontologies with varying size represented by the number of universities (X-axis) is given here. The reasoning tasks are Consistency Checking (CC) and Realisation (RT). The dashed line, parallel to X-axis represents the time-out (90 min).

to check the performance of a reasoner on a set of language constructs that could possibly be of interest to the user. Another possibility is to check the performance of a reasoner under a given set of constraints, for example, time-taken or memory consumed during the reasoning process. We propose the following two extensions that address the last two requirements.

3.1 Customizable Selection of OWL 2 Language Constructs

The idea behind this approach is inspired by OntoBench [6]. It provides a web interface to the users to select the constructs of their choice and generates an ontology according to the selected constructs. However, the primary purpose of OntoBench is to test the reasoner coverage and not to evaluate the performance and the scalability of the reasoners. We propose a customizable TBox generator. Here, along with the choice of constructs, the user can specify the individual count for each selected construct. Note that, instead of continuing with OntoBench's general approach and its naming convention for the generated classes and properties, we propose to use our benchmark's university ontology. The axioms generated by OntoBench lack interconnections that are necessary to test the efficiency of the reasoners. However, a university ontology consists of concepts that describe a university (college, department, faculty, etc.) and the relationships between them. Thus, the axioms in the generated ontology would have sufficient interconnections for performance evaluation. Moreover, a domain-specific ontology improves the readability of large ontologies.

In this approach, we first create a bucket for each OWL 2 construct. The TBox axioms of our benchmark already cover all the major OWL 2 constructs. We start putting these axioms into the bucket of the construct involved. For example, axioms like `Faculty` $\sqsubseteq$ $\exists$`worksFor.College` would be put into the bucket of *existential restriction*. If the user chooses *existential restriction*, then axioms from this bucket would be picked. Note that, along with each axiom, some related axioms also get generated. For example, along with `Faculty` $\sqsubseteq$ $\exists$`worksFor.College`, axioms such as `Faculty` $\sqsubseteq$ `Employee`, `College` $\sqsubseteq$ `University`, and `University` $\sqsubseteq$ `Organization` would also get generated. If the count exceeds the number of axioms in the bucket, then the axioms can be repeated with a different naming convention such as `Faculty_1` $\sqsubseteq$ $\exists$`worksFor.College_1`, and `Employee_1` $\sqsubseteq$ $\exists$`worksFor.Organization_1`. Furthermore, we would also consider the interactions and the interplay between the axioms. It is possible that reasoners can handle a particular set of axioms generated for a certain set of constructors well. However, with the same set of constructors, a different set of axioms could cause a blow up for the reasoner because these axioms result in interactions that did not occur in the previous case. Therefore, we plan to generate different sets of axioms (more than one ontology) for the user-selected constructors. Since the bucket for each constructor consists of several axioms, it is feasible to generate multiple ontologies. However, there are some design decisions that still need to be made, such as, should both `Faculty` and `Faculty_1` be a subclass of `Employee` or should they be subclass of `Employee` and `Employee_1` respectively. This needs to be investigated so that the benchmark does not generate a large number of unnecessary side axioms.

3.2 Generating Ontologies based on their Hardness

We define three levels (easy, medium, and hard) to categorize an ontology. These are defined based on reasoner performance metrics such as the time taken and the memory consumed while reasoning over an ontology. The first step is to determine the distinguishing *features* of an ontology that can help in defining the three levels. For this purpose, we plan to make use of existing work on determining different aspects of the size and structural characteristics of an ontology that affect the reasoner's performance [4]. Several ontology *features* such as the number of named entities (classes, properties), class unions, depth of class and property hierarchy, the ratio of object and data properties, etc., have been studied for their impact on the reasoning performance. A large number of ontologies from different repositories (ORE dataset[10], AberOWL[11]) would be run on different OWL 2 reasoners for the three reasoning tasks: classification, consistency checking, and realisation. The results obtained in terms of the time taken and the memory consumed, serve as the basis to cluster these ontologies under the three levels, i.e., easy, medium, and hard. Each cluster would have a different range of values for the ontology *features*.

[10] http://doi.org/10.5281/zenodo.18578
[11] http://aber-owl.net/ontology/

The OWL2Bench users would be provided with an option to choose any one of the hardness levels, along with the total number of axioms in the ontology. Based on these inputs, and the values of ontology *features* associated with that hardness level, OWL2Bench automatically generates an ontology that can be used to benchmark the OWL 2 reasoners. The approach used would be similar to Section 3.1. The only difference is that instead of directly considering the type and the number of each construct from the user as inputs, we try to balance the axioms in such a way that the values of the *features* in the generated ontology comply with the ranges specified in that particular hardness level.

References

1. Glimm, B., Horrocks, I., Motik, B., S., G., Wang, Z.: HermiT: An OWL 2 Reasoner. Journal of Automated Reasoning. 53(3), 245–269 (2014)
2. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Journal of Web Semantics. 3(2-3), 158–182 (2005)
3. Hitzler, P., Krötzsch, M., Parsia, B., F. Patel-Schneider, P., Rudolph, S.: OWL 2 Web Ontology Language Profiles (Second Edition) (2012), https://www.w3.org/TR/owl2-primer/
4. Kang, Y.B., Krishnaswamy, S., Sawangphol, W., Gao, L., Li, Y.F.: Understanding and improving ontology reasoning efficiency through learning and ranking. Information Systems **87**, 101412 (2020)
5. Kazakov, Y., Krötzsch, M., Simančík, F.: The Incredible ELK. Journal of Automated Reasoning. 53(1), 1–61 (2014)
6. Link, V., Lohmann, S., F., H.: OntoBench: Generating Custom OWL 2 Benchmark Ontologies. In: International Semantic Web Conference. pp. 122–130 (2016)
7. Ma, L., Yang, Y., Qiu, Z .and Xie, G., Pan, Y., Liu, S.: Towards a Complete OWL Ontology Benchmark. In: The Semantic Web: Research and Applications. pp. 125–139. Springer Berlin Heidelberg (2006)
8. Singh, G., Bhatia, S., Mutharaju, R.: OWL2Bench: A Benchmark for OWL 2 Reasoners. In: The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, ISWC 2020. Lecture Notes in Computer Science, Springer (2020)
9. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics. 5(2), 51–53 (2007)
10. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. Journal of Web Semantics. 27, 78–85 (2014)
11. Yadav, R.K., Singh, G., Mutharaju, R., Bhatia, S.: Towards a Concurrent Approximate Description Logic Reasoner. In: Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas). CEUR Workshop Proceedings, vol. 2456, pp. 145–148 (2019)