OWL2Bench: A Benchmark for OWL 2 Reasoners

Gunjan Singh[1], Sumit Bhatia[2], and Raghava Mutharaju[1]

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

Abstract. There are several existing ontology reasoners that span a wide spectrum in terms of their performance and the expressivity that they support. In order to benchmark these reasoners to find and improve the performance bottlenecks, we ideally need several real-world ontologies that span the wide spectrum in terms of their size and expressivity. This is often not the case. One of the potential reasons for the ontology developers to not build ontologies that vary in terms of size and expressivity, is the performance bottleneck of the reasoners. To solve this chicken and egg problem, we need high quality ontology benchmarks that have good coverage of the OWL 2 language constructs, and can test the scalability of the reasoners by generating arbitrarily large ontologies. We propose and describe one such benchmark named OWL2Bench. It is an extension of the well-known University Ontology Benchmark (UOBM). OWL2Bench consists of the following - TBox axioms for each of the four OWL 2 profiles (EL, QL, RL, and DL), a synthetic ABox axiom generator that can generate axioms of arbitrary size, and a set of SPARQL queries that involve reasoning over the OWL 2 language constructs. We evaluate the performance of six ontology reasoners and two SPARQL query engines that support OWL 2 reasoning using our benchmark. We discuss some of the performance bottlenecks, bugs found, and other observations from our evaluation.

Resource Type: Benchmark
License: Apache License 2.0
Code and Queries: https://github.com/kracr/owl2bench

Keywords: OWL 2 benchmark · ontology benchmark · OWL2Bench · reasoner benchmark

1 Introduction

OWL 2 [6] has different profiles, namely OWL 2 EL, OWL 2 QL, OWL 2 RL, OWL 2 DL, and OWL 2 Full that vary in terms of their expressivity and reasoning complexity. The first three profiles are tractable fragments of OWL 2 having polynomial reasoning time. On the other hand, reasoning over OWL 2 DL ontologies has a complexity of N2EXPTIME and OWL 2 Full is undecidable. Several thousands of ontologies that belong to these OWL 2 profiles are available across repositories such as the NCBO Bioportal[3], AgroPortal[4], and the ORE datasets [12]. In order to compare features and benchmark the performance of different reasoners, one could use a subset of these existing ontologies. However, such an approach is inflexible as most real-world ontologies cover only a limited set of OWL constructs and arbitrarily large, and complex ontologies are seldom available that can be used to test the limits of systems being benchmarked. A synthetic benchmark, on the other hand, offers the flexibility to test various aspects of the system by changing the configuration parameters (such as size and complexity). In particular, it is hard to answer the following questions without a benchmark.

i) Does the reasoner support all the possible constructs and their combinations of a particular OWL 2 profile?
ii) Can the reasoner handle large ontologies? What are its limits?
iii) Can the reasoner handle all types of queries that involve reasoning?
iv) What is the effect of any particular construct, in terms of number or type, on the reasoner performance?

Unless a reasoner can answer these questions, ontology developers will not have the confidence to build large and complex ontologies. Without these ontologies, it will be hard to test the performance and scalability of the reasoner. So an ontology benchmark can fill this gap and help the developers in building better quality reasoners and ontologies.

There are some existing benchmarks such as LUBM [5], UOBM [9], BSBM [3], SP^{2}Bench [15], DBpedia [10], and OntoBench [8]. Some of these are based on RDF (BSBM, SP^{2}Bench, DBpedia) or an older version of OWL (LUBM, UOBM), and those that cover OWL 2 are limited in scope (OntoBench). We propose an OWL 2 benchmark, named OWL2Bench, that focuses on the coverage of OWL 2 language constructs, tests the scalability, and the query performance of OWL 2 reasoners. The main contributions of this work are as follows.

- TBox axioms for each of the four OWL 2 profiles (EL, QL, RL, and DL). They are developed by extending UOBM's university ontology. These axioms are helpful to test the inference capabilities of the reasoners.
- An ABox generator that can generate axioms of varying size over the aforementioned TBox. This is useful for testing the scalability of the reasoners.
- A set of 22 SPARQL queries spanning different OWL 2 profiles that require reasoning in order to answer the queries. These queries also enable benchmarking of SPARQL query engines that support OWL 2 reasoning.

[3]https://bioportal.bioontology.org/
[4]http://agroportal.lirmm.fr/

Six reasoners, namely, ELK [7], HermiT [4], JFact[5], Konclude [17], Openllet[6], and Pellet [16] were evaluated using OWL2Bench on three reasoning tasks (consistency checking, classification, realisation). SPARQL queries were used to evaluate the performance of Stardog[7] and GraphDB[8]. The results of our evaluation are discussed in Section 4.

2 Related Work

There has been limited work on developing benchmarks for OWL 2 reasoners. On the other hand, there are several established benchmarks for RDF query engines and triple stores such as LUBM [5], BSBM [3], SP^{2}Bench [15], WatDiv [1], DBpedia benchmark [10], and FEASIBLE [14]. These benchmarks have been discussed comprehensively in [13]. Since the focus of our work is not on benchmarking the RDF query engines and triple stores, we do not discuss these benchmarks any further.

LUBM [5] provides an ontology for the university domain that covers a subset of OWL Lite constructs, a dataset generator to generate variable size instance data, and a set of 14 SPARQL queries. In the generated dataset, different universities lack necessary interlinks that do not sufficiently test the scalability aspect of the ontology reasoners. To test the scalability of reasoners, we need to increase the size of the generated data. In the case of LUBM, this is achieved by increasing the number of universities. So, if the benchmark lacks necessary interlinks across different universities, the generated instance data would result in multiple isolated graphs rather than a connected large graph. Reasoning over a connected large graph is significantly harder than that on multiple isolated small graphs. Thus, interlinks are necessary to reveal the inference efficiency of the reasoners on scalable datasets. University Ontology Benchmark (UOBM) [9] is an extension of LUBM that has been designed to overcome some of the drawbacks of LUBM. It covers most of the OWL Lite and OWL DL constructs to test the inference capability of the reasoners. Additional classes and properties have been added to create interlinks among the universities. But, UOBM does not support the OWL 2 profiles.

OntoBench [8] covers all the constructs and profiles supported by OWL 2. The primary purpose of OntoBench is to test the coverage of the reasoners rather than their scalability. It provides a web interface[9] for the users to choose the OWL 2 language constructs which are then used to generate an ontology. Thus, OntoBench overcomes the inflexibility of the other static benchmarks. However, it does not support the generation of ABox axioms.

JustBench [2] is a benchmark for ontology reasoners in which the performance is analyzed based on the time taken by the reasoners to generate justifications for

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

Table 1. A comparison of the state-of-the-art benchmarks with OWL2Bench. C indicates coverage in terms of OWL constructs/SPARQL features, S indicates scalability, R indicates the three reasoning tasks (consistency checking, classification, and realisation)

Benchmark	**Supported Profile**	**Evaluated System(s)**	**Evaluation Type**
LUBM	OWL Lite (Partial)	RDFS and OWL Lite Reasoners	C, S
UOBM	OWL Lite and OWL 1 DL	OWL Lite and OWL 1 DL Reasoners	C, S
SP^{2}Bench	RDFS	RDF Stores	C, S
BSBM	RDFS	RDF Stores	C, S
DBpedia	RDFS/OWL (Partial)	RDF Stores	C, S
OntoBench	All OWL 2 Profiles	Ontology Visualization Tool	C
ORE Framework	OWL 2 EL and DL	OWL 2 Reasoners	R
OWL2Bench	All OWL 2 Profiles	OWL 2 Reasoners and SPARQL Query Engines	C, S, R

the entailments. This makes the benchmark independent of the OWL versions and profiles. JustBench does not generate any data (TBox or ABox axioms). So the benchmark, by itself, cannot be used to test the scalability of the reasoners.

Other than the aforementioned benchmarks, there also exists an open-source java based ORE benchmark framework[10] which was a part of OWL Reasoner Evaluation (ORE) Competition [11,12]. The competition was held to evaluate the performance of OWL 2 complaint reasoners over several different OWL 2 EL and OWL 2 DL ontologies. But, the performance evaluation in the context of varying sizes of an ontology was not considered. The ORE competition corpus can be used with the framework for reasoner evaluation. The framework evaluates the reasoners on three reasoning tasks, namely, consistency checking, classification, and realisation. The framework does not cover the evaluation of the SPARQL query engines (with OWL reasoning support) in terms of the coverage of constructs and scalability.

Comparison of OWL2Bench with some of the state-of-the-art benchmarks is provided in Table 1. There is no benchmark that is currently available that can test the OWL 2 reasoners in terms of their coverage, scalability, and query performance. We address this shortcoming by proposing OWL2Bench.

[10]https://github.com/ykazakov/ore-2015-competition-framework

3 OWL2Bench Description

OWL2Bench can be used to benchmark three aspects of the reasoners - support for OWL 2 language constructs, scalability in terms of ABox size, and the query performance. It consists of three major components, a fixed TBox for each OWL 2 profile, an ABox generator that can generate ABox of varying size with respect to the corresponding TBox, and a fixed set of SPARQL queries that can be run over the generated ontology (combination of TBox and ABox).

3.1 TBox

We built the TBox for each of the four OWL 2 profiles (EL, QL, RL, and DL), by extending UOBM because it has support for OWL Lite and OWL DL profiles[11], along with a mechanism to generate ABox axioms and has a set of SPARQL queries for the generated ABox. UOBM consists of concepts that describe a university (college, department, course, faculty, etc.) and the relationships between them. The TBoxes in OWL2Bench are created by following the steps listed below.

i) The axioms from UOBM that belong to at least one of the OWL 2 profiles are added to the TBox of the respective profiles and those that did not fit into the corresponding OWL 2 profile due to the syntactic restrictions are restructured to fit into that profile.
ii) There are several other language constructs from across the four OWL 2 profiles that are not covered by the axioms from UOBM. In such cases, either the UOBM axioms are enriched with additional constructs or new classes and properties are introduced, and appropriate axioms are added to the respective TBoxes.
iii) In order to create a more interconnected graph structures when compared to UOBM, additional axioms that link different universities are created. For example, object properties `hasCollaborationWith`, and `hasAdvisor` connect a `Person` from a `Department` with people across different departments of the same `College`, and across different colleges of the same `University`, or across different universities.

The hierarchy among some of the classes, including the relations between them, is shown in Figure 1. All the four TBoxes of OWL2Bench consist of classes such as `University`, `College`, `CollegeDiscipline`, `Department`, `Person`, `Program`, and `Course`. They are related to each other through relationships such as `enrollFor`, `teachesCourse`, and `offerCourse`.

Some of the extensions to the axioms, classes, and properties made to UOBM for each OWL 2 profile are listed in Table 2. The complete list of extensions is available on GitHub[12]. Apart from ObjectSomeValuesFrom, the table contains only those OWL 2 constructs that are absent in UOBM. We included

[11] Profiles of OWL 1
[12] https://github.com/kracr/owl2bench

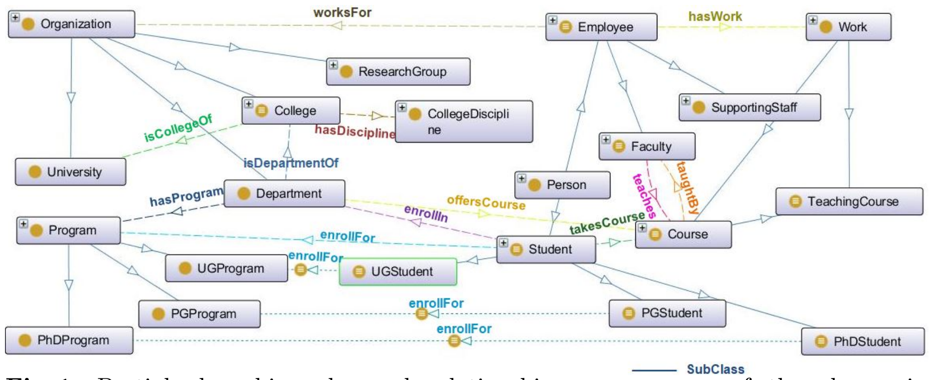

Fig. 1. Partial class hierarchy and relationship among some of the classes in OWL2Bench. The colored labeled (dashed) edges represent different object properties. The unlabeled edges represent the subclass relation.

ObjectSomeValuesFrom to illustrate the syntactic restrictions imposed on the usage of different constructs in each OWL 2 profile. For example, we renamed the UOBM's `UndergraduateStudent` class to `UGStudent`, added a new class `UGProgram` and a property `enrollFor`, and redefined the class `UGStudent` (UG student is an undergraduate student who enrolls for any undergraduate program). The class `UGStudent` is added to all the profiles and the definition changes slightly (given below) depending on the OWL 2 profile.

- In OWL 2 EL, $\texttt{UGStudent} \equiv \texttt{Student} \sqcap \exists\texttt{enrollFor.UGProgram}$
- In OWL 2 QL, $\texttt{UGStudent} \sqsubseteq \texttt{Student} \sqcap \exists\texttt{enrollFor.UGProgram}$ (existential restrictions are not allowed in the subclass expression)
- In OWL 2 RL, $\texttt{Student} \sqcap \exists\texttt{enrollFor.UGProgram} \sqsubseteq \texttt{UGStudent}$ (existential restrictions are not allowed in the superclass expression)
- In OWL 2 DL, $\texttt{UGStudent} \equiv \texttt{Student} \sqcap =1\texttt{enrollFor.UGProgram}$ (since qualified exact cardinalities are supported, we could make the axiom more expressive by writing it using exact cardinality)

3.2 ABox

ABox axioms are generated by OWL2Bench based on two user inputs, the number of universities and the OWL 2 profile (EL, QL, RL, DL) of interest. The instance data that is generated complies with the schema defined in the TBox of the selected profile. The size of the instance data depends on the number of universities. The steps to generate the ABox are listed below.

i) Instances (class assertion axioms) for the `University` class are generated and their number is equal to the number of universities specified by the user.
ii) For each `University` class instance, instances for `College`, `Department`, as well as for all the related classes are generated.

Table 2. Examples of TBox axioms from OWL2Bench. Axioms follow the syntactic restriction of the OWL 2 profiles.

OWL 2 EL	
ObjectPropertyChain	$\texttt{worksFor} \circ \texttt{isPartOf} \sqsubseteq \texttt{isMemberOf}$
ObjectSomeValuesFrom	$\texttt{UGStudent} \equiv \texttt{Student} \sqcap \exists\texttt{enrollFor.UGProgram}$
ReflexiveObjectProperty	ReflexiveObjectProperty(`knows`)
ObjectHasSelf	$\texttt{SelfAwarePerson} \equiv \texttt{Person} \sqcap \exists\texttt{knows.Self}$
OWL 2 QL	
ObjectSomeValuesFrom	$\texttt{UGStudent} \sqsubseteq \texttt{Student} \sqcap \exists\texttt{enrollFor.UGProgram}$
IrreflexiveObjectProperty	IrreflexiveObjectProperty(`isAdvisedBy`)
AsymmetricObjectProperty	AsymmetricObjectProperty(`isAffiliatedOrganisationOf`)
OWL 2 RL	
ObjectSomeValuesFrom	$\texttt{Student} \sqcap \exists\texttt{enrollFor.UGProgram} \sqsubseteq \texttt{UGStudent}$
ObjectAllValuesFrom	$\texttt{WomenCollege} \sqsubseteq \texttt{College} \sqcap \forall\texttt{hasStudent.Women}$
ObjectMaxCardinality	$\texttt{LeisureStudent} \sqsubseteq \texttt{Student} \sqcap \leq 1\ \texttt{takesCourse.Course}$
OWL 2 DL	
ObjectAllValuesFrom	$\texttt{WomenCollege} \equiv \texttt{College} \sqcap \forall\texttt{hasStudent.Women}$
ObjectExactCardinality	$\texttt{UGStudent} \equiv \texttt{Student} \sqcap =1\texttt{enrollFor.UGProgram}$
ObjectMaxCardinality	$\texttt{LeisureStudent} \equiv \texttt{Student} \sqcap \leq 1\ \texttt{takesCourse.Course}$
ObjectMinCardinality	$\texttt{PeopleWithManyHobbies} \equiv \texttt{Person} \sqcap \geq 3\ \texttt{likes.Interest}$
DisjointDataProperties	DisjointDataProperties(`firstName lastName`)
Keys	HasKey(`Person hasID`)
DisjointObjectProperties	DisjointObjectProperties(`likes dislikes`)
DisjointUnion	$\texttt{CollegeDiscipline} \equiv \texttt{Engineering} \sqcup \texttt{Management} \sqcup \texttt{Science} \sqcup \texttt{FineArts} \sqcup \texttt{HumanitiesAndSocialScience}$
DisjointClasses	$\texttt{Engineering} \sqcap \texttt{Management} \sqcap \texttt{Science} \sqcap \texttt{FineArts} \sqcap \texttt{HumanitiesAndSocialScience} \sqsubseteq \bot$

iii) Property assertion axioms are created using these instances. For example, an object property `isDepartmentOf` links a `Department` instance to a `College` instance. Similarly, a data property `hasName` is used to connect a department name to a `Department` instance.

iv) The number of instances of each class (other than `University`) and the number of connections between all the instances are selected automatically and randomly from a range specified in the configuration file. This range (maximum and minimum values of the parameters) can be modified to change the size of the generated ABox as well as to control the density (number of connections between different instances). Moreover, the output ontology format can also be specified in the configuration file. By default, the generated ontology format is RDF/XML.

Since we focus on testing the scalability of the reasoners OWL2Bench, generates (by default) approximately 50,000 ABox axioms for one university and it goes up to 14 million for 200 universities. Table 3 shows the number of TBox

Table 3. The number of TBox axioms, along with the number of ABox axioms generated by OWL2Bench is given here.

Type		**OWL 2 Profile**			
		EL	**QL**	**RL**	**DL**
Classes		131	131	134	132
Object Properties		82	85	85	86
Data Properties		12	12	12	13
TBox Axioms		703	738	780	793
	1 University	50,131	50,125	50,131	50,127
	2 Universities	99,084	99,078	99,084	99,080
	5 Universities	325,412	325,406	325,412	325,408
ABox	**10 Universities**	711,021	711,015	711,021	711,017
Axioms	**20 Universities**	1,380,428	1,380,422	1,380,428	1,380,424
	50 Universities	3,482,119	3,482,113	3,482,119	3,482,115
	100 Universities	7,260,070	7,260,064	7,260,070	7,260,066
	200 Universities	14,618,828	14,618,822	14,618,828	14,618,824

axioms and the ABox axioms that are generated by OWL2Bench based on the number of universities.

3.3 Queries

While the generated A-Box and T-Box can be used to benchmark capabilities and performance of various reasoners, OWL2Bench also provides twenty-two SPARQL queries as part of the benchmark. These queries have been designed with the aim of evaluating SPARQL engines that support reasoning (such as Stardog). Each query has a detailed description that captures the intent of the query and the OWL 2 language constructs that the SPARQL engine needs to recognize and process accordingly. Each query makes use of language constructs that belong to at least one OWL 2 profile. Due to the lack of space, we have not included all the queries in the Appendix. They are available on GitHub[12].

4 Experiments and Discussions

In this section, we use our benchmark to compare the reasoning and querying performance of six reasoners and two SPARQL query engines. Note that the aim of our experiments is not to present an exhaustive analysis of all the existing reasoners and SPARQL query engines. Instead, we chose a representative subset to demonstrate the utility of OWL2Bench. 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 also pave the way for further research in the development of reasoners and query engines.

All our evaluations ran on a server with an AMD Ryzen Threadripper 2990WX 32-Core Processor and 128GB of RAM. The server ran on a 5.3.0-46-generic #38 18.04.1-Ubuntu SMP operating system. We use Java 1.8 and OWL API 5.1.11 in the experiments. We set the heap space to 24GB. We run our benchmark on the reasoners ELK 0.4.3 [7], HermiT 1.3.8.1 [4], JFact 5.0.0[5], Konclude 0.6.2 [17], Openllet 2.6.4[6], and Pellet 2.3.6 [16]. ELK supports OWL 2 EL profile and the rest of them are OWL 2 DL reasoners. We use GraphDB 9.0.0[8] and Stardog 7.0.2[7] for running the SPARQL queries. GraphDB supports SPARQL queries and OWL 2 QL, OWL 2 RL profiles (but not OWL 2 EL and OWL 2 DL). Stardog's underlying reasoner is Pellet and hence it supports ontologies of all the profiles including OWL 2 DL. Note that only those reasoners and SPARQL query engines were considered for evaluation that offered full reasoning support with respect to the OWL 2 profiles.

4.1 Evaluating OWL Reasoners

We compare the six ontology reasoners with respect to the time taken for performing three major reasoning tasks, namely, consistency checking, classification, and realisation. We use OWL2Bench to generate ABox axioms for 1, 2, 5, 10, 20, 50, 100, and 200 Universities for the OWL 2 EL, QL, RL, and DL profiles. Table 3 reports the size of each of these datasets. Table 4 reports the time taken by the reasoners on these datasets. The time-out was set to 90 minutes. We report the average time taken in 5 independent runs for each reasoning task.

We used the OWL API to connect to HermiT, JFact, Pellet, Openllet, and run them. Since the reasoner implementations do quite a bit of work in the reasoner constructor, we included this time as well while calculating the time taken for each reasoning task. For the other two reasoners (ELK and Konclude), parsing and preprocessing times are included by default when run from the command line. Moreover, ELK also includes the ontology loading time in its reported time and ignores some of the axioms (such as Object Property Range, Data Property Assertion, Data Property Domain, and Range, Functional and Equivalent Data Property, Self Restriction and Same Individual) during the preprocessing stage. Since ELK is an OWL 2 EL reasoner, we compare its performance only in the EL profile.

JFact has performed the worst on all the reasoning tasks across all the four OWL 2 profiles. It timed out for every reasoning task except in the OWL 2 QL profile. Even for the QL profile, JFact performed the worst even on the smallest ontology and timed-out when the size of the dataset increased. Konclude outperformed all the other reasoners in terms of time taken for all the three reasoning tasks (except for OWL 2 DL datasets). It was able to complete the reasoning task even when most of the other reasoners had an error or timed out. Although Konclude is faster than most of the reasoners, it requires a lot of memory and thus as the size of the dataset grew larger (100 and 200 universities) it threw a memory error. It is interesting to observe that the ontologies generated by OWL2Bench are reasonably challenging and most of the reasoners were unable to handle the classification and realisation tasks on these ontologies.

Table 4. Time taken in seconds for the three reasoning tasks: Consistency Checking (CC), Classification (CT) and Realisation (RT) by the reasoners on the ontologies from OWL2Bench. j.h.s is Java Heap Space Error, g.c is Garbage Collection Overhead limit error, m.e is Memory Error, and t.o is Timed Out.

Profile	**Task**	**Reasoner**	**No. of Universities**							
			1	**2**	**5**	**10**	**20**	**50**	**100**	**200**
EL	**CC**	**ELK**	1.27	2.47	8.43	12.57	28.32	70.48	106.63	197.68
		HermiT	8.87	35.65	665.12	t.o	t.o	t.o	t.o	t.o
		JFact	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o
		Konclude	0.88	1.92	7.67	17.70	35.59	89.68	m.e	m.e
		Openllet*	-	-	-	-	-	-	-	-
		Pellet	2.65	6.93	84.75	608.32	1876.00	t.o	t.o	t.o
	CT	**ELK**	1.50	2.50	8.51	12.56	30.93	70.77	106.83	177.59
		HermiT	110.13	639.56	t.o	t.o	t.o	t.o	t.o	g.c
		JFact	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o
		Konclude	0.88	1.92	7.68	17.70	35.60	89.69	m.e	m.e
		Openllet	9.53	19.41	627.02	1803.05	t.o	t.o	t.o	t.o
		Pellet	11.46	37.72	335.14	t.o	t.o	t.o	t.o	t.o
	RT	**ELK**	1.60	2.69	8.97	13.42	32.45	74.03	114.48	195.26
		HermiT	t.o	t.o	t.o	t.o	t.o	j.h.s	g.c	g.c
		JFact	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o
		Konclude	0.90	1.95	7.77	17.93	36.03	90.70	m.e	m.e
		Openllet	13.88	38.18	281.23	1829.93	t.o	t.o	t.o	t.o
		Pellet	6.08	18.99	234.74	t.o	t.o	t.o	t.o	t.o
QL	**CC**	**HermiT**	1.34	3.23	6.56	14.71	29.37	65.81	151.40	392.60
		JFact	15.77	53.98	656.75	3207.62	t.o	t.o	t.o	t.o
		Konclude	0.66	1.41	5.70	13.56	26.72	66.29	m.e	m.e
		Openllet	0.93	1.69	4.81	11.87	25.12	51.23	121.03	335.12
		Pellet	0.96	1.60	5.09	10.93	21.20	51.26	121.866	303.18
	CT	**HermiT**	1.70	3.37	10.01	23.85	51.61	132.50	300.04	901.10
		JFact	29.88	93.76	935.12	4222.124	t.o	t.o	t.o	t.o
		Konclude	0.66	1.42	5.70	13.57	26.73	66.30	m.e	m.e
		Openllet	1.21	2.20	5.79	12.63	25.84	55.00	127.26	357.696
		Pellet	1.39	2.64	6.43	12.34	24.88	58.87	131.00	340.31
	RT	**HermiT**	1.70	3.37	10.01	23.85	51.61	132.50	300.04	901.10
		JFact	87.85	364.94	3776.16	t.o	t.o	t.o	t.o	t.o
		Konclude	0.68	1.44	5.79	13.75	27.08	67.20	m.e	m.e
		Openllet	1.95	3.16	6.85	14.43	27.27	65.22	149.11	426.81
		Pellet	1.15	1.91	5.62	12.53	24.63	61.91	141.13	350.85
RL	**CC**	**HermiT**	147.17	1782.14	t.o	t.o	t.o	j.h.s	t.o	g.c
		JFact	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o
		Konclude	0.75	1.62	6.50	15.08	31.49	74.28	m.e	m.e
		Openllet	9.60	46.11	614.67	2910.94	t.o	t.o	t.o	t.o
		Pellet	3.67	12.51	118.30	467.38	1657.55	t.o	t.o	t.o
	CT	**HermiT**	2768.19	t.o	t.o	t.o	t.o	t.o	t.o	g.c
		JFact	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o
		Konclude	0.76	1.63	6.57	15.08	31.50	74.29	m.e	m.e
		Openllet	13.49	45.20	631.06	2776.39	t.o	t.o	t.o	t.o
		Pellet	8.68	24.49	179.12	1069.02	t.o	t.o	t.o	t.o
	RT	**HermiT**	209.31	2938.86	t.o	t.o	t.o	j.h.s	t.o	g.c
		JFact	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o
		Konclude	0.78	1.66	6.68	15.32	31.91	75.29	m.e	m.e
		Openllet	10.29	49.88	675.60	2996.07	t.o	t.o	t.o	t.o
		Pellet	4.28	15.06	144.64	640.09	3046.57	t.o	t.o	t.o
DL	**CC**	**HermiT**	138.41	2206.51	t.o	t.o	t.o	t.o	t.o	g.c
		JFact	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o
		Konclude	78.36	m.e	m.e	m.e	m.e	m.e	m.e	m.e
		Openllet*	-	-	-	-	-	-	-	-
		Pellet	859.61	149.63	1977.04	t.o	t.o	t.o	t.o	t.o
	CT	**HermiT**	t.o	t.o	t.o	t.o	t.o	t.o	t.o	g.c
		JFact	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o
		Konclude	m.e	m.e	m.e	m.e	m.e	m.e	m.e	m.e
		Openllet	824.51	4013.19	t.o	t.o	t.o	t.o	t.o	g.c
		Pellet	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o
	RT	**HermiT**	t.o	t.o	t.o	t.o	t.o	t.o	t.o	g.c
		JFact	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o
		Konclude	m.e	m.e	m.e	m.e	m.e	m.e	m.e	m.e
		Openllet	t.o	t.o	t.o	t.o	t.o	t.o	t.o	g.c
		Pellet	t.o	t.o	t.o	t.o	t.o	t.o	t.o	t.o

Table 5. Inconsistent results (in seconds) from Openllet for OWL 2 EL and OWL 2 DL ontologies generated by OWL2Bench for the consistency checking reasoning task.

Profile	**No. of**	**Iteration**				
	Univ.	**1**	**2**	**3**	**4**	**5**
EL	**1**	7.60	10.48	7.63	7.11	0.87
	2	53.96	5.7	37.98	43.13	2.59
	5	14.08	622.66	579.30	710.91	8.97
	10	45.20	32.42	39.31	52.90	2236.29
	20	t.o	170.55	t.o	t.o	117.09
	50	725.94	733.84	t.o	t.o	668.14
	100	3155.08	t.o	4408.96	t.o	t.o
	200	t.o	t.o	t.o	t.o	t.o
DL	**1**	1.48	4.09	t.o	t.o	1.57
	2	2.57	3.22	3.21	2.46	2.00
	5	t.o	t.o	t.o	t.o	t.o
	10	13.77	t.o	t.o	t.o	13.76
	20	t.o	t.o	t.o	t.o	t.o
	50	t.o	97.98	1790.08	480.09	t.o
	100	t.o	t.o	t.o	170.39	178.38
	200	428.91	t.o	t.o	342.18	t.o

For the OWL 2 EL profile, the two concurrent reasoners, ELK and Konclude performed exceptionally well when compared to HermiT, Openllet, and Pellet. Their performance deteriorated with increase in the size of the ontology. On the smaller ontologies (up to 5 universities), Konclude performed better than ELK in all the reasoning tasks. But for the larger ontologies, ELK (a profile specific reasoner) performed better than Konclude (an all profile reasoner) in terms of runtime and memory. We did not include Openllet consistency results in Table 4 since we noticed some inconsistencies in the results obtained over different iterations. These are reported in Table 5. We observed two types of inconsistencies. Across the different runs on the same ontology, there is a huge variation in the runtime. For example, for 10 universities in the OWL 2 EL profile, Openllet's runtime varied from 32.42 seconds to 2236.29 seconds and for 20 universities, it varied from 117.09 seconds to a timeout (greater than 90 minutes). The second inconsistency that we observed is that Openllet takes more time or times out on a smaller ontology but completes the same reasoning task on a larger ontology. A case in point are the column values of "Iteration 2" in the EL profile across the universities (Table 5). The inconsistency in the results was reported to the openllet support[13].

[13] https://github.com/Galigator/openllet/issues/50

Table 6. Load times (in seconds) for Stardog and GraphDB for OWL2Bench data for four profiles, and for 1, 5, and 10 universities. d.s.e denotes Disk Space Error. n.a denotes reasoner not applicable for the corresponding OWL 2 Profile.

	1 University				**5 Universities**				**10 Universities**			
	EL	**QL**	**RL**	**DL**	**EL**	**QL**	**RL**	**DL**	**EL**	**QL**	**RL**	**DL**
GraphDB	n.a	9.26	2619.53	n.a	n.a	30.66	d.s.e	n.a	n.a	59.30	d.s.e	n.a
Stardog	7.17	6.79	6.86	6.99	8.42	7.97	8.27	7.96	10.08	10.00	10.01	9.60

Since Openllet is an extension of Pellet, we ran the same experiments as in Table 5 using Pellet to check for the inconsistencies in the results. We observed only one among the two inconsistencies, which is that consistency checking of a larger ontology sometimes takes more time than for a smaller ontology. Since both these ontologies belong to the same OWL 2 profile and the same TBox was used, the results are surprising.

Most of the reasoners (except JFact) worked well on all the reasoning tasks in the OWL 2 QL profile. Pellet/Openllet perform the best in terms of the runtime, but as expected, the performance of all the reasoners deteriorate with increase in the size of the ontology. The performance of Konclude is comparable to that Pellet and Openllet, except that it was not able to handle ontologies that are very large (100 and 200 universities).

In the OWL 2 RL profile, most reasoners could not handle the larger ontologies. The performance of Pellet and Openllet was comparable in all the other profiles. But in OWL 2 RL, Pellet's performance was significantly better when compared to Openllet. In the case of OWL 2 DL profile, Openllet completed the classification reasoning task for 1 and 2 universities. HermiT and Pellet were able to complete the consistency checking task only (for 2 and 5 universities, respectively). Other than that, most of the reasoners timed out or had memory related errors even for small ontologies (1 and 2 universities).

4.2 Evaluating SPARQL Engines

We evaluated the twenty-two SPARQL queries from OWL2Bench on GraphDB and Stardog. We compare these two systems in terms of their loading time and query response time. We use the ontologies generated by OWL2Bench for 1, 5, and 10 universities (Table 3) for the experiments.

Loading time comparison: Table 6 reports the time taken (averaged over 5 independent runs) by the two SPARQL engines. Stardog consistently outperforms GraphDB and is able to load the data an order of magnitude faster than GraphDB. This is because GraphDB performs materialization during load time and thus the size of the GraphDB repository is much more than the Stardog repository. We observed that GraphDB could not load the data from OWL 2 RL ontology for 5 and 10 universities. It ended up using more than 2.5 TB of disk space and gave an insufficient disk space error after about 6 hours. This indicates that GraphDB may not be able to handle even medium sized ontologies (around 300k axioms). This issue has been reported to the GraphDB support team.

Table 7. Time taken (in seconds) by Stardog and GraphDB for different SPARQL queries. ABox is generated for 1, 5, and 10 universities for the four OWL 2 profiles. *o.w* indicates that the system did not produce the expected result due to Open World Assumption. *t.o* indicates time out (10 minutes). *n.a* denotes Not Applicable because datasets could not be loaded for querying. 'x' indicates that the particular query is not applicable to an OWL 2 profile.

Query	**EL**			**QL**						**RL**						**DL**		
	Stardog			**GraphDB**			**Stardog**			**GraphDB**			**Stardog**			**Stardog**		
	1	**5**	**10**	**1**	**5**	**10**	**1**	**5**	**10**	**1**	**5**	**10**	**1**	**5**	**10**	**1**	**5**	**10**
Q1	2.95	1.69	3.71	0.50	0.80	1.10	0.91	2.14	3.77	x	x	x	x	x	x	t.o	t.o	t.o
Q2	1.59	4.27	7.60	x	x	x	x	x	x	0.20	n.a	n.a	3.58	14.63	29.01	t.o	t.o	t.o
Q3	0.13	0.22	0.27	x	x	x	x	x	x	0.40	n.a	n.a	0.21	0.65	1.17	t.o	t.o	t.o
Q4	0.34	0.92	1.32	x	x	x	x	x	x	0.70	n.a	n.a	0.21	0.71	0.78	t.o	t.o	t.o
Q5	0.13	0.20	0.11	x	x	x	x	x	x	0.60	n.a	n.a	0.13	0.16	0.18	t.o	t.o	t.o
Q6	1.84	2.08	4.00	x	x	x	x	x	x	x	x	x	x	x	x	t.o	t.o	t.o
Q7	x	x	x	0.20	0.30	0.40	0.62	1.39	1.90	0.20	n.a	n.a	0.20	1.44	1.71	t.o	t.o	t.o
Q8	x	x	x	0.20	0.30	0.40	0.14	0.14	0.14	0.30	n.a	n.a	0.14	0.16	0.17	t.o	t.o	t.o
Q9	x	x	x	o.w	o.w	o.w	o.w	o.w	o.w	o.w	n.a	n.a	o.w	o.w	o.w	t.o	t.o	t.o
Q10	x	x	x	0.30	0.40	0.70	0.22	0.40	0.62	0.60	n.a	n.a	0.26	0.40	0.42	t.o	t.o	t.o
Q11	x	x	x	0.70	0.20	0.20	0.33	0.54	0.91	0.20	n.a	n.a	0.44	0.96	1.36	t.o	t.o	t.o
Q12	x	x	x	x	x	x	x	x	x	0.20	n.a	n.a	205.44	654.00	t.o	t.o	t.o	t.o
Q13	x	x	x	x	x	x	x	x	x	o.w	n.a	n.a	o.w	o.w	o.w	t.o	t.o	t.o
Q14	x	x	x	x	x	x	x	x	x	o.w	n.a	n.a	o.w	o.w	o.w	t.o	t.o	t.o
Q15	x	x	x	x	x	x	x	x	x	0.10	n.a	n.a	0.12	0.16	0.21	t.o	t.o	t.o
Q16	x	x	x	x	x	x	x	x	x	0.10	n.a	n.a	0.14	0.09	0.19	t.o	t.o	t.o
Q17	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	t.o	t.o	t.o
Q18	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	t.o	t.o	t.o
Q19	0.09	0.19	0.28	0.20	0.20	0.20	0.11	0.18	0.28	0.20	n.a	n.a	0.09	0.20	0.28	t.o	t.o	t.o
Q20	0.23	0.48	0.84	0.20	0.30	0.90	0.13	0.31	0.47	0.30	n.a	n.a	99.08	t.o	t.o	t.o	t.o	t.o
Q21	0.24	0.37	0.71	0.10	0.20	0.20	0.15	0.35	0.39	0.20	n.a	n.a	0.86	0.65	1.38	t.o	t.o	t.o
Q22	0.18	0.22	0.36	0.10	0.20	0.20	0.19	0.26	0.37	0.20	n.a	n.a	0.17	0.11	0.31	t.o	t.o	t.o

Query Runtime Comparison: We use the twenty-two SPARQL queries from OWL2Bench to compare the query runtime performance of GraphDB and Stardog. Note that, despite the availability of a number of SPARQL engines such as

Virtuoso[14], Blazegraph[15], and RDF-3X[16], we chose these two SPARQL engines because of their support for OWL 2 reasoning, albeit with different expressivity. The time-out for the query execution is 10 minutes for each query. Table 7 summarizes the query runtimes averaged over 5 independent runs for the two systems. The queries involving constructs from OWL 2 EL and OWL 2 DL have been ignored for GraphDB since it does not support those two profiles. Since GraphDB does inferencing at load time, it is able to answer all the OWL 2 QL and OWL 2 RL related queries in a fraction of a second for even larger datasets. Stardog, on the other hand, performs reasoning at run time and thus timed out for Q12 on the OWL 2 RL 10 universities ontology and Q20 on OWL 2 RL with 5 and 10 universities. For Stardog, we observed that it could not handle any of the queries related to the OWL 2 DL profile (Q1 to Q22) across the three ontologies (1, 5, and 10 Universities). It could, however, handle most of the queries for other profiles. Due to the open world assumption, we do not get the desired results for Q9, Q13, and Q14 since they involve cardinalities, complement, and universal quantifiers.

5 Conclusions and Future Work

We presented an ontology benchmark named OWL2Bench. The focus of our benchmark is on testing the coverage, scalability, and query performance of the reasoners across the four OWL 2 profiles (EL, QL, RL, and DL). To that end, we extended UOBM to create four TBoxes for each of the four OWL 2 profiles. Our benchmark also has an ABox generator and comes with a set of twenty-two SPARQL queries that involve reasoning. We demonstrated the utility of our benchmark by evaluating six reasoners and two SPARQL engines using our benchmark. Inconsistencies in the results were observed in some of the cases and these have been already reported to the appropriate support teams.

We plan to extend this work by making the TBox generation more configurable, i.e., users can select the particular language constructs (across all the four OWL 2 profiles) that they are interested in benchmarking. Another extension that we plan to work on is to provide an option to the users to choose the desired hardness level (easy, medium, and hard) of the ontology with respect to the reasoning time and OWL2Bench will then generate such an ontology. The hardness of an ontology can be measured in terms of the reasoning runtime and the memory needed to complete the reasoning process.

Acknowledgement. The first and the third author would like to acknowledge the partial support received from the Infosys Center for Artificial Intelligence at IIIT-Delhi.

[14]https://virtuoso.openlinksw.com/
[15]https://blazegraph.com/
[16]https://code.google.com/archive/p/rdf3x/

References

1. Aluç, G., Hartig, O., Özsu, M.T., Daudje, K.: Diversified Stress Testing of RDF Data Management Systems. In: The Semantic Web – ISWC 2014. pp. 197–212. Springer International Publishing, Cham (2014)
2. Bail, S., Parsia, B., Sattler, U.: JustBench: A Framework for OWL Benchmarking. In: The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I. Lecture Notes in Computer Science, vol. 6496, pp. 32–47. Springer (2010)
3. Christian, B., Andreas, S.: The Berlin SPARQL Benchmark. International Journal on Semantic Web and Information Systems. 5, 1–24 (2009)
4. Glimm, B., Horrocks, I., Motik, B., S., G., Wang, Z.: HermiT: An OWL 2 Reasoner. Journal of Automated Reasoning. 53(3), 245–269 (2014)
5. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Journal of Web Semantics. 3(2-3), 158–182 (2005)
6. 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/
7. Kazakov, Y., Krötzsch, M., Simančík, F.: The Incredible ELK. Journal of Automated Reasoning. 53(1), 1–61 (2014)
8. Link, V., Lohmann, S., F., H.: OntoBench: Generating Custom OWL 2 Benchmark Ontologies. In: International Semantic Web Conference. pp. 122–130 (2016)
9. 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, Berlin, Heidelberg (2006)
10. Morsey, M., Lehmann, J., Auer, S., Ngomo, A.C.: DBpedia SPARQL Benchmark–Performance Assessment with Real Queries on Real Data. In: The Semantic Web – ISWC 2011. pp. 454–469 (2011)
11. Parsia, B., Matentzoglu, N., Gonçalves, R.S., Glimm, B., Steigmiller, A.: The OWL Reasoner Evaluation (ORE) 2015 Resources. In: The Semantic Web – ISWC 2016. pp. 159–167. Springer International Publishing, Cham (2016)
12. Parsia, B., Matentzoglu, N., Gonçalves, R.S., Glimm, B., Steigmiller, A.: The OWL Reasoner Evaluation (ORE) 2015 Competition Report. Journal of Automated Reasoning. 59(4), 455–482 (2017)
13. Sakr, S., Wylot, M., Mutharaju, R., Le Phuoc, D., Fundulaki, I.: Linked Data - Storing, Querying, and Reasoning. Springer (2018)
14. Saleem, M., Mehmood, Q., Ngonga Ngomo, A.: FEASIBLE: A Feature-Based SPARQL Benchmark Generation Framework. In: The Semantic Web - ISWC 2015. pp. 52–69. Springer International Publishing, Cham (2015)
15. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP2Bench: A SPARQL Performance Benchmark, pp. 371–393. Springer Berlin Heidelberg (2010)
16. 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)
17. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. Journal of Web Semantics. 27, 78–85 (2014)

Appendix

Some of the OWL2Bench SPARQL Queries are listed below.

Q1. SELECT DISTINCT ?x ?y WHERE { ?x :knows ?y }
Description: Find the instances who know some other instance.
Construct Involved: knows is a Reflexive Object Property.
Profile: EL, QL, DL

Q2. SELECT DISTINCT ?x ?y WHERE { ?x :isMemberOf ?y }
Description: Find Person instances who are member (Student or Employee) of some Organization.
Construct Involved: ObjectPropertyChain
Profile: EL, RL, DL

Q6. SELECT DISTINCT ?x ?y WHERE { ?x rdf:type :SelfAwarePerson }
Description: Find all the instances of class SelfAwarePerson. Self Aware person is a Person who knows themselves.
Construct Involved: ObjectHasSelf
Profile: EL, DL

Q8. SELECT DISTINCT ?x ?y WHERE { ?x :isAffiliatedOrganizationOf ?y }
Description: Find the Affiliations of all the Organizations.
Construct Involved: isAffiliatedOrganizationOf is an Asymmetric Object Property. Domain(Organization), Range(Organization).
Profile: QL, RL, DL

Q11. SELECT DISTINCT ?x ?y WHERE { ?x :isAdvisedBy ?y }
Description: Find all the instances who are advised by some other instance.
Construct Involved: isAdvisedBy is an Irreflexive Object Property. Domain(Person), Range(Person)
Profile: QL, RL, DL

Q17. SELECT DISTINCT ?x WHERE {?x rdf:type :UGStudent}
Description: Find all the instances of class UGStudent. UGStudent is a Student who enrolls in exactly one UGProgram.
Construct Involved: ObjectExactCardinality
Profile: DL