Towards A Complete OWL Ontology Benchmark

Li Ma, Yang Yang, Zhaoming Qiu, Guotong Xie, Yue Pan, Shengping Liu

IBM China Research Laboratory, Building 19, Zhongguancun Software Park,
ShangDi, Beijing, 100094, P.R. China
{malli, yangyy, qiuzhaom, xieguot, panyue, liusp}@cn.ibm.com

Abstract. Aiming to build a complete benchmark for better evaluation of existing ontology systems, we extend the well-known Lehigh University Benchmark in terms of inference and scalability testing. The extended benchmark, named University Ontology Benchmark (UOBM), includes both OWL Lite and OWL DL ontologies covering a complete set of OWL Lite and DL constructs, respectively. We also add necessary properties to construct effective instance links and improve instance generation methods to make the scalability testing more convincing. Several well-known ontology systems are evaluated on the extended benchmark and detailed discussions on both existing ontology systems and future benchmark development are presented.

1 Introduction

The rapid growth of information volume in World Wide Web and corporate intranets makes it difficult to access and maintain the information required by users. Semantic Web aims to provide easier information access based on the exploitation of machine-understandable metadata. Ontology, a shared, formal, explicit and common understanding of a domain that can be unambiguously communicated between human and applications, is an enabling technology for Semantic Web. W3C has recommended two standards for publishing and sharing ontologies on the World Wide Web: Resource Description Framework (RDF) [3] and Web Ontology Language (OWL) [4,5]. OWL facilitates greater machine interpretability of web content than that supported by RDF and RDF Schema (RDFS) by providing additional vocabulary along with formal semantics. That is, OWL has more powerful expressive capability which is required by real applications and is thus the current research focus. In the past several years, some ontology toolkits, such as Jena [23], KAON2 [22] and Sesame [14], had been developed for ontologies storing, reasoning and querying. A standard and effective benchmark to evaluate existing systems is much needed.

1.1 Related Work

In 1998, Description Logic (DL) community developed a benchmark suite to facilitate comparison of DL systems [18,19]. The suite included concept satisfiability tests, synthetic TBox classification tests, realistic TBox classification tests and synthetic

ABox tests. Although DL is the logic foundation of OWL, the developed DL benchmarks are not practical to evaluate ontology systems. DL benchmark suite tested complex inference, such as satisfiability tests of large concept expressions, and did not cover realistic and scalable ABox reasoning due to poor performance of most systems at that time. This is significantly far away from requirements of Semantic Web and ontology based enterprise applications. Tempich and Volz [16] conducted a statistical analysis on more than 280 ontologies from DAML.ORG library and pointed out that ontologies vary tremendously both in size and their average use of ontological constructs. These ontologies are classified into three categories, taxonomy or terminology style, description logic style and database schema-like style. They suggested that Semantic Web benchmarks have to consist of several types of ontologies.

SWAT research group of Lehigh University [9,10,20] made significant efforts to design and develop Semantic Web benchmarks. Especially in 2004, Guo et al. developed Lehigh University Benchmark (LUBM) [9,10] to facilitate the evaluation of Semantic Web tools. The benchmark is intended to evaluate the performance of ontology systems with respect to extensional queries over a large data set that conforms to a realistic ontology. The LUBM appeared at a right time and was gradually accepted as a standard evaluation platform for OWL ontology systems. More recently, Lehigh Bibtex Benchmark (LBBM) [20] was developed with a learned probabilistic model to generate instance data. According to Tempich and Volz's classification scheme [16], the LUBM is to benchmark systems processing ontologies of description logic style while the LBBM is for systems managing database schema-like ontologies. Different from the LUBM, the LBBM represents more RDF-style data and queries. By participating in a number of enterprise application development projects (e.g., metadata and master data management) with IBM Integrated Ontology Toolkit [12], we learned that RDFS is not expressive enough for enterprise data modeling and OWL is more suitable than RDFS for semantic data management. The primary objective of this paper is to extend the LUBM for better benchmarking OWL ontology systems.

OWL provides three increasingly expressive sublanguages designed for use by specific communities of users [4]: OWL Lite, OWL DL, and OWL Full. Implementing complete and efficient OWL Full reasoning is practically impossible. Currently, OWL Lite and OWL DL are research focuses. As a standard OWL ontology benchmark, the LUBM has two limitations. Firstly, it does not completely cover either OWL Lite or OWL DL inference. For example, inference on cardinality and allValueFrom restrictions cannot be tested by the LUBM. In fact, the inference supported by this benchmark is only a subset of OWL Lite. Some real ontologies are more expressive than the LUBM ontology. Secondly, the generated instance data may form multiple relatively isolated graphs and lack necessary links between them. More precisely, the benchmark generates individuals (such as departments, students and courses) taking university as a basic unit. Individuals from a university do not have relations with individuals from other universities (here, we mean the relations intentionally involved in reasoning.) Therefore, the generated instance is grouped by university. This results in multiple relatively separate university graphs. Apparently, it is less reasonable for scalability tests. Inference on a complete and huge graph is substantially harder than that on multiple isolated and small graphs. In summary, the LUBM is weaker in measuring infer-

ence capability as well as less reasonable to generate big data sets for measuring scalability.

1.2 Contributions

In this paper, we extend the Lehigh University Benchmark so that it could better provide both OWL Lite and OWL DL inference tests (except TBox with cyclic class definition. Hereinafter, OWL Lite or OWL DL complete is understood with this exception) on more complicated instance data sets. The main contributions of the paper are as follows.

- The extended Lehigh University Benchmark, named University Ontology Benchmark (UOBM), is OWL DL complete. Two ontologies are generated to include inference of OWL Lite and OWL DL, respectively. Accordingly, queries are constructed to test inference capability of ontology systems.
- The extended benchmark generates instance data sets in a more reasonable way. The necessary links between individuals from different universities make the test data form a connected graph rather than multiple isolated graphs. This will guarantee the effectiveness of scalability testing.
- Several well-known ontology systems are evaluated on the extended benchmark and conclusions are drawn to show the state of arts.

The remainder of the paper is organized as follows. Section 2 analyzes and summarizes the limitations of the LUBM and presents the UOBM, including ontology design, instance generation, query and answer construction. Section 3 reports the experimental results of several well-known ontology systems on the UOBM and provides detailed discussions. Section 4 concludes this paper.

2 Extension of Lehigh University Benchmark

This section provides an overview of the LUBM and analyzes its limitations as a standard evaluation platform. Based on such an analysis, we further propose methods to extend the benchmark in terms of ontology design, instance generation, query and answer construction.

2.1 Overview of the LUBM

The LUBM is intended to evaluate the performance of ontology systems with respect to extensional queries over a large data set that conforms to a realistic ontology. It consists of an ontology for university domain, customizable and repeatable synthetic data, a set of test queries, and several performance metrics. The details of the benchmark can be found in [9,10]. As a standard benchmark, the LUBM itself has two limitations. Firstly, it covers only part of inference supported by OWL Lite and OWL DL. Table 1 tabulates all OWL Lite and OWL DL language constructs which are inference-related as well as those supported by the LUBM (in underline).

Table 1. OWL Constructs Supported by the LUBM

OWL Lite

RDF Schema Features:
- **rdfs:subClassOf**
- **rdfs:subPropertyOf**
- **rdfs:domain**
- **rdfs:range**

Property Characteristics:
- **ObjectProperty**
- **DatatypeProperty**
- **inverseOf**
- **TransitiveProperty**
- SymmetricProperty
- FunctionalProperty
- InverseFunctional Property

Class Intersection:
- **IntersectionOf**

Property Restrictions:
- allValuesFrom
- **someValuesFrom**

Restricted Cardinality:
- minCardinality (only 0 or 1)
- maxCardinality (only 0 or 1)
- cardinality (only 0 or 1)

(In)Equality:
- equivalentClass
- equivalentProperty
- sameAs
- differentFrom
- AllDifferent
- distinctMembers

OWL DL

Class Axioms:
- oneOf, dataRange
- disjointWith
- equivalentClass (applied to class expressions)
- rdfs:subClassOf (applied to class expressions)

Boolean Combinations of Class Expressions:
- unionOf
- complementOf
- intersectionOf

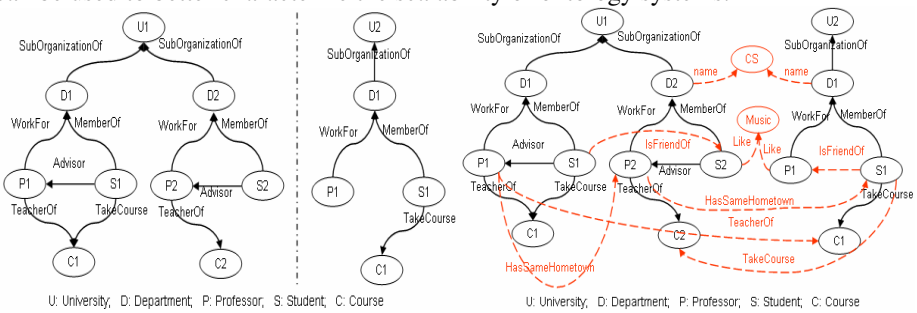
Arbitrary Cardinality:
- minCardinality
- maxCardinality
- cardinality

Filler Information:
- hasValue

The above table shows clearly that the LUBM's university ontology only uses a small part of OWL Lite and OWL DL constructs (the used constructs are in underline) and thus covers only part of OWL inference. That is, it cannot exactly and completely evaluate an ontology system in terms of inference capability. In fact, some constructs excluded by LUBM's ontology, such as allValuesFrom, cardinality, oneOf and SymmetricProperty, are very useful for expressive data modeling in practice. For example, using construct hasValue, we can define class "basketBallLover" whose property "like" has a value of "basketBall". We found that the LUBM's ontology is less expressive than some real ontologies. With the increasing uses of ontologies in practical applications, more and more complex ontologies will appear. Obviously, more constructs (hence more inference requirements) should be included for system evaluation.

Another limitation of the LUBM is that the generated instance data may form multiple relatively isolated graphs and lacks necessary links between them for scalability testing. Figure 1(a) shows a simplified example of the LUBM generated instance (the real instance may include more universities and more departments in a university). We can see from this figure that there are two relatively independent university graphs, and two relatively independent department graphs in the same university. Such kind of data is less challenging for scalability testing. As is well known, to evaluate the scalability of a system, we generally observe the system performance changes with the increasing size of the data. Here, the increase of the testing data means that more universities will be generated. Due to the relative independence of the data of different universities, the performance changes of an ontology system on an Relational DBMS (currently, most ontology repositories are on top of RDBMS) with such data sets will be determined to a large extent by the underlying database. This cannot really reveal the inference efficiency of an ontology system, considering the fact that inference on a complete and huge RDF graph is significantly harder than that on multiple isolated and small graphs with comparable number of classes and properties. The underlying reason leading to such a case is that the instance generator of the LUBM creates data using university as a basic unit and does not intentionally construct individuals and relationships across universities. Therefore, we will enhance the instance generator of

the LUBM to generate instances in a more practical way. As shown in Figure 1(b), crossing-university and crossing-department relations will be added to form a more complicated graph. For instance, professor can teach course in different departments and universities, and students can have friends from different universities. In the LUBM, it is possible that two persons from different universities graduate from the same university (by property degreeFrom). Here, our intention is to add more links between universities and the links should be involved in reasoning, which is challenging for scalability tests. Compared with the graph in Fig 1(a), the graph in Fig. 1(b) can be used to better characterize the scalability of ontology systems.



(a) Original graph (b) Enriched graph

Fig. 1. Instance Graph Enrichment of the LUBM

2.2 University Ontology Benchmark (UOBM)

Based on our analysis on the LUBM, we can conclude that LUBM is insufficient to evaluate the inference capability and less effective to reflect the scalability of an ontology system. We build University Ontology Benchmark (UOBM) based on the LUBM to solve these two problems. Figure 2 gives an overview of the UOBM. It consists of three major components, ontology selector, instance generator and queries and answers analyzer. These core components are detailed in the following subsections.

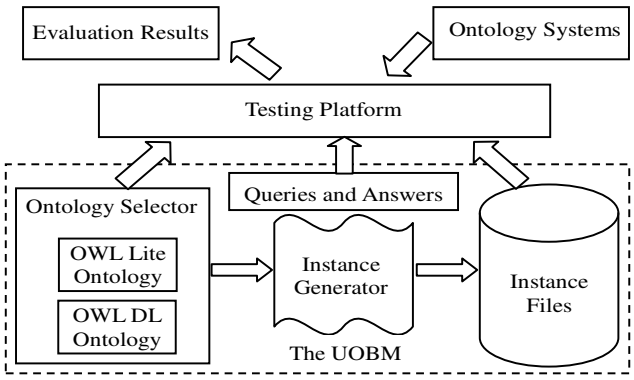


Fig. 2. Overview of the UOBM

2.2.1 Ontology Selector

Different from the original LUBM, the UOBM includes both OWL Lite and OWL DL ontologies. That is, one ontology includes all language constructs of OWL Lite, and another one covers all OWL DL constructs. The user can specify which ontology will be used for evaluation according to specific requirements. As Table 1 shows, a number of OWL constructs are absent in the LUBM. For those absent constructs, we newly define corresponding classes and properties in the UOBM. Table 2 lists our major extensions for OWL Lite and OWL DL ontologies, respectively. Classes and properties corresponding to the constructs in the table are represented in W3C's OWL language abstract syntax [5]. Due to space limitation, some classes and properties, namespace of URIs and enumerated values in oneOf classes are not listed there.

Table 2. Class and Property Extensions of the UOBM

OWL Lite	
allValueFrom	Class(GraduateStudent, complete intersectionOf(restriction(takesCourse, someValueFrom(Thing)), restriction(takesCourse, allValue-From(GraduateCourse))))
minCardinality	Class(PeopleWithHobby, restriction(like, minCardinality(1)))
EquivalentProperty	EquivalentProperty(like, love)
EquivalentClass	EquivalentClass(Person, Humanbeing)
SymmetricProperty	ObjectProperty (isFriendOf, Symmetric, domain(Person), range(Person))
TransitiveProperty	ObjectProperty (hasSameHomeTownWith, Symmetric|Transitive, do-main(Person), range(Person))
FunctionalProperty	ObjectProperty(isTaughtBy, Functional, domain(Course), range(Faculty))
InverseFunctional Property	ObjectProperty(isHeadOf, InverseFunctional, domain(Person), range(Organization))
OWL DL	
disjointWith	DisjointClasses(Man, Woman)
oneOf	Class(Science, oneOf(Physics, Mathematics ….)) Class(Engineer, oneOf(Electical_Engineer, Chemical_Engineer…)) …
unionOf	Class(Person, unionOf(Man, Woman)) Class(AcademicSubject, unionOf(Science, Engineer, FineArts, Humanitie-sAndSocial))
complementOf	Class(NonScienceStudnet, complementOf(restriction(hasMajor, someVal-ueFrom(Science)))) Class(WomanCollege, complete intersectionOf(College, retriction (hasStu-dent, allValueFrom(complementOf(Man)))))
intersectionOf	Class(SwimmingFan, complete intersectionOf(Person, restriction (isCrazy-About, hasValue(Swimming)))
hasValue	Class(BasketBallLover, restriction(like, value(BasketBall))) Class(TennisFan, restriction(isCrazyAbout, value(Tennis)))…
minCandinality	Class(PeopleWithMultipleHobbies, restriction(like, minCardinality(3)))
maxCandinality	Class(LeisureStudent, intersactionOf(UndergraduateStudent, restric-tion (takesCourse, maxCardinality(2))))
Candinality	Class(PeopleWith2Hobbies, restriction(like, Cardinality(2)))
EquivalentClass	EquivalentClass(TeachingAssistant, complete intersectionOf(Person, restric-tion (teachingAssistantOf, someValueFrom(Course))))

Table 3 shows a comparison between the LUBM and the UOBM in terms of the number of classes, properties and individuals per university. The number of classes and properties used to define ABox are denoted in the bracket. This means that some

classes and properties are only used to define class and property hierarchies in TBox and not used to directly restrict individuals. But users can issue queries using such classes and properties constraints. Individuals in TBox are used to define oneOf and hasValue restrictions. We can see from the table that the UOBM can generate much larger and more complex instance graph. More important is that it covers all OWL Lite and OWL DL constructs. An effective evaluation on the benchmark will help researchers to figure out more problems and promote the development of ontology systems. Note that the number of instances shown in Table 3 (e.g., No. of statements per univ.) is assessed based on parameters used in [9] and used in our experiments presented in next section, respectively.

Table 3. Comparison of the LUBM and the UOBM

Benchmark	The LUBM	The UOBM	
		OWL Lite	OWL DL
No. of Classes	43 (22)	51 (41)	69 (59)
No. of Datatype Property	7 (3)	9 (5)	9 (5)
No. of Object Property	25(14)	34(24)	34 (24)
No. of Individuals in TBox	0	18	58
No. of Statements per University	90,000 – 110,000	210,000 – 250,000	220,000 – 260,000
No. of Individuals per University	8,000 – 15,000	10,000 – 20,000	10,000 – 20,000

2.2.2 Instance Generator

Instance generator automatically and randomly creates instances according to user-specified ontology (OWL Lite or OWL DL). Also, the user can specify the size of the generated instance data by setting the number of universities to be constructed. Compared with the LUBM, we extend following properties to link individuals from different departments and universities. As a result, the UOBM will enable the construction of a complicated connected graph instead of multiple relatively-isolated graphs.

- ObjectProperty (isFriendOf, Symmetric, domain(Person), range(Person))
- ObjectProperty(hasSameHomeTownWith, Symmetric|Transitive, domain(Person), range(Person))
- ObjectProperty(takesCourse, domain(Student))
- ObjectProperty (hasMajor, domain(Student), range(AcademicSubject))
- ObjectProperty (like, domain(Person), range(Interest))
  EquivalentProperties(love, like)
- ObjectProperty (isCrazyAbout, super(like), domain(Person), range(Interest))

Instance generator can be configured to generate data sets for specific evaluation. Some important parameters for building a connected graph are listed below.

- Specify ontology, OWL Lite or OWL DL (**parameter for TBox configuration**)
- Specify the probability that a student takes courses of other departments and universities, and the range of the number of courses a student takes.
- Specify the probability that a person has the same hometown with those from other departments and universities. (Affect the ratio of transitive properties as well)

- Specify the probability that a person has friends of other departments and universities, and the range of the number of friends a person has.
- Specify the probability that a university has woman college, and the range of the number of students.
- Specify the probability that a person has some hobbies.

2.2.3 Queries and Answers Analyzer

A set of queries are constructed to evaluate the inference capability and scalability of an ontology system. Queries are designed based on two principles: 1) Queries need search and reasoning across universities so that the scalability of a system can be better characterized. In the original LUBM, some queries are evaluated only on specific universities and departments regardless of the increasing size of the testing data. This results mainly from lacks of links between different universities. 2) Each query supports at least a different type of OWL inference. By this way, if a query cannot be correctly answered, we can easily identify which kind of inference is not well supported. The test queries are listed in appendix with detailed explanations.

Given queries and randomly generated test data, we have to find corresponding correct answers in order to compute completeness and soundness of the inference. The original LUBM does not explicitly provide a method to generate correct results. Our current scheme is to import all statements into an RDBMS such as DB2 or MySQL, and then manually translate each query into SQL queries to retrieve all correct results. It is feasible because we know inference required by every query and can use a DL reasoner for TBox inference and build SQL queries on the inferred TBox for ABox inference and retrieval. Also, we use some tricks for SQL query rewriting, for example, naming convention of instances. The manual translation method has been written into a standalone application in the benchmark. It is convenient to run the application to obtain answer sets.

Using the UOBM, the user can follow a simple approach for performance evaluation of ontology systems. Firstly, the user selects an ontology (OWL Lite or OWL DL) to generate corresponding instances. Then, using the built-in query translation method, the user can obtain correct query results in advance. Finally, based on the selected ontology, generated instances, test queries and correct answers, load time, query response time, inference completeness and soundness of a system can be easily computed. Currently, the UOBM is publicly available at [12].

3. Evaluation of Ontology Systems and Discussions

In this section, we use the UOBM to evaluate several well-known ontology systems and discuss problems deserving further research work based on experimental results. ***This work is not intended to make a complete evaluation for existing OWL ontology systems. From our preliminary experiments, we hope to find some critical problems to promote the development of OWL ontology systems as well as figure out more issues needed to be considered in a complete benchmark.***

3.1 Target Systems and Experiments Setting

In [9], Guo et al. conducted a quantitative evaluation on the LUBM for four knowledge base systems, Sesame's persistent storage and main memory version [14,15], OWLJessKB [13], and DLDB-OWL [8]. They used data loading time, repositories sizes, query response time, query completeness and soundness as evaluation metrics. Experimental results showed that, as a whole, DLDB-OWL outperformed other systems on large-scale data sets. OWLIM [18] is a newly developed high performance repository and is packaged as a Storage and Inference Layer (SAIL) for Sesame. Recently, IBM released its Integrated Ontology Development Toolkit [12], including an ontology repository (named Minerva), EMF based Ontology Definition Metamodel and a workbench for ontology editing. Here, we will evaluate these persistent ontology repositories, DLDB-OWL, OWLIM (version 2.8.2) and Minerva (version 1.1.1).

We will have a brief look at these systems so that we can understand the experimental results better. DLDB-OWL [8] is a repository for processing, storing, and querying large amounts of OWL data. Its major feature is the extension of a relational database system with description logic inference capabilities. It uses the DL reasoner to precompute class subsumption and employs relational views to answer extensional queries based on the implicit hierarchy that is inferred. Minerva [12] completely implements the inference supported by Description Logic Program (DLP), an intersection of Description Logic and Horn Logic Program. Its highlight is a hybrid inference method which uses Racer or Pellet DL reasoner to obtain implicit subsumption among classes and properties and adopts DLP logic rules for instance inference. Minerva designs the schema of the back-end database completely according to the DLP logic rules to support efficient inference. OWLIM is a high-performance semantic repository, wrapped as a Storage and Inference Layer for the Sesame RDF database. OWLIM uses Ontotext's TRREE to perform forward-chaining rule reasoning. The reasoning and query are conducted in-memory. At the same time, a reliable persistence strategy assures data preservation, consistency and integrity.

Our evaluation method is similar to the one used in [9]. Here, 6 test data sets are generated, Lite-1, Lite-5, Lite-10, DL-1, DL-5 and DL-10, where the alphabetic string indicates the type of the ontology and is followed by an integer indicating the number of universities. Each university contains about 20 departments and over 210,000 statements. The most complex and largest data set, DL-10, includes over 2,200,000 statements. Test queries are listed in the appendix of the paper, where 13 queries for OWL Lite tests and 3 more for OWL DL tests. Experiments are conducted on a PC with Pentium IV CPU of 2.66 GHz and 1G memory, running Windows 2000 professional with Sun Java JRE 1.4.2 (JRE 1.5.0 for OWLIM) and Java VM memory of 512M. The following three metrics [9] are used for comparison.

- Load time. The time for loading a data set into memory or persistence storage. It includes reasoning time since some systems do TBox or ABox inference at load time.
- Query response time. The time for issuing a query, obtaining the result set and traversing the results sequentially.
- Completeness and soundness. Completeness measures the recall of a system's answer to a query and soundness measures its precision.

3.2 Evaluation of OWL Ontology Systems

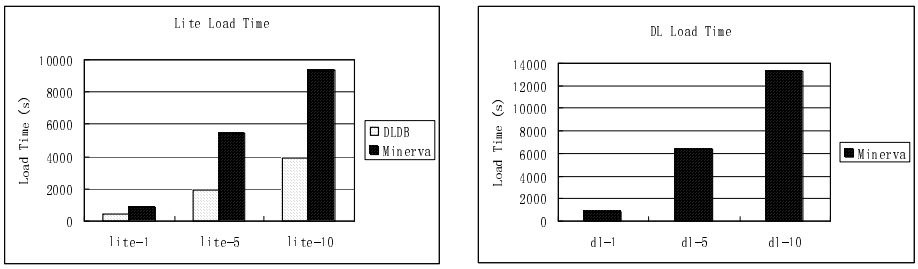


Fig. 3. Load Time Comparison

Figure 3 shows load time of Minerva and DLDB-OWL (hereinafter, DLDB denotes DLDB-OWL). Since OWLIM takes only 29 seconds to load Lite-1, it is too small to plot it in the figure. OWLIM is substantially faster than other two systems as reasoning is done in memory. But, OWLIM cannot complete forward-chaining inference on other data sets due to memory limitation. There are no results for DLDB on DL data sets as an exception was thrown out when loading OWL DL files. DLDB is faster than Minerva to load data sets because it does not conduct ABox materialization at load time. In fact, Minerva's performance on loading and reasoning on OWL data is high, only about 2.5 hours for over 2.2M triples from Lite-10 data set. Its storage schema provides effective support for inference at load time.

OWLIM does inference in memory. Therefore, it can answer queries more quickly than DLDB and Minerva. But its scalability is relatively poor. In most cases, Minerva outperforms DLDB in terms of query response time. The reason is that Minerva does all inference at load time and directly retrieves results using SQL queries at query time, whereas DLDB uses class views which are built based on inferred class hierarchy at load time to retrieve instances at query time. DLDB's view query (a view is equivalent to a query in relational database.) needs to execute union operations in runtime which is more expensive than select operations on pre-built index in most cases. The last three subfigures in Fig. 4 show the scalability of DLDB on Lite data sets and that of Minerva on both Lite and DL data sets, respectively. We observe that for most queries, the query time of DLDB grows dramatically with the increase of the size of the data set. But Minerva scales much better than DLDB. For some queries, such as queries 13 and 15, the query time of Minerva is almost zero and does not change too much since there are few or no results. One may find that Minerva's query time for query 8 increases significantly on DL-10. The reason is that there are a large number of results. Since the query time includes time to traverse results sequentially (the original LUBM uses such a definition as well), it can be affected by the number of results.

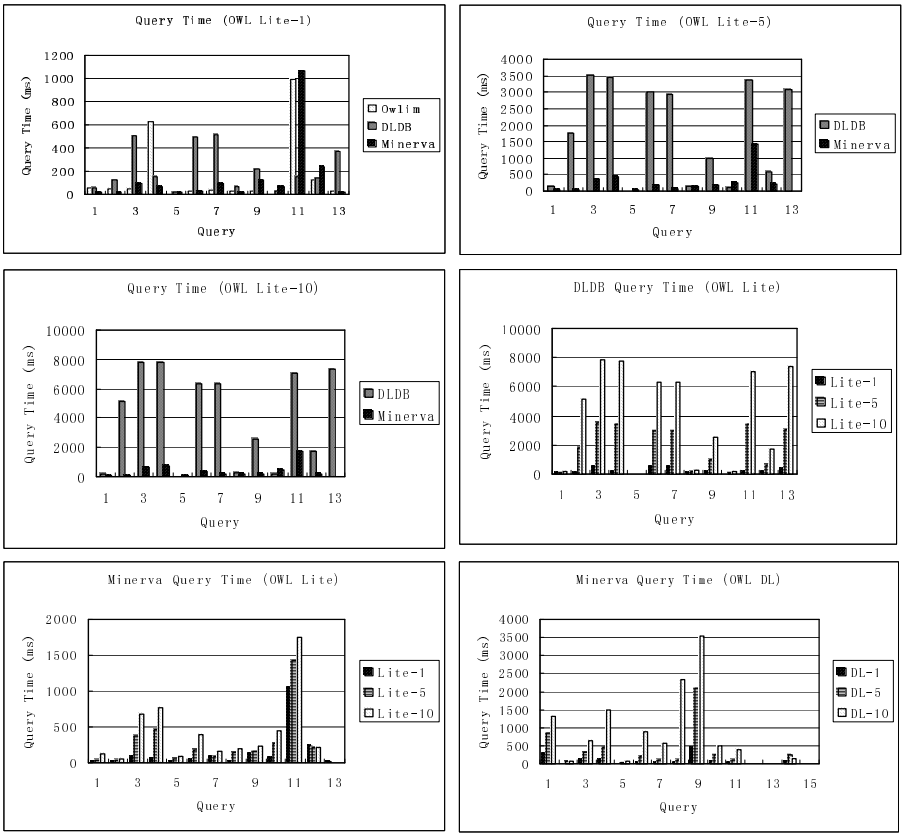


Fig. 4. Query Response Time Comparison

Our experiments have confirmed that all three systems are sound, i.e., the precision is 1. Table 4 shows query completeness results. Compared with previous version, OWLIM 2.8.2 can answer all queries correctly. In this new release of OWLIM, more rules are added and inference is made configurable. As is known, OWL-Lite and OWL-DL reasoning cannot be implemented only by rules. That is, OWLIM currently conducts partial OWL DL TBox inference. This is different from DLDB and Minerva which depend on a DL reasoner for TBox inference. Coincidentally, the UOBM does not contain a query that needs subsumption inference not covered by existing OWLIM rules. This indicates that the UOBM should add more complex class definition and corresponding instances and queries. The inference capability of DLDB is relatively weak and it gives 100% complete answers to only 3 queries. Minerva is able to completely and correctly process 12 out of 13 queries. Inference on minCardinality needed by query 13 is not currently supported in Minerva. These three persistence systems use rules for ABox inference. How to support more ABox rules on large-scale data sets which cannot be fit into memory directly deserves more efforts.

Table 4. Query Completeness Comparison

Query		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
OWLIM	Lite-1	1	1	1	1	1	1	1	1	1	1	1	1	1	NA	NA
DLDB	Lite-1	1	0.82	1	1	0	0	0	0	0	0.83	0	0.2	0.51	NA	NA
	Lite-5	1	0.81	1	1	0	0	0	0	0	0.59	0	0.12	0.57	NA	NA
	Lite-10	1	0.81	1	1	0	0	0	0	0	0.87	0	0.26	0.53	NA	NA
Minerva	Lite-1	1	1	1	1	1	1	1	1	1	1	1	1	0.67	NA	NA
	Lite-5	1	1	1	1	1	1	1	1	1	1	1	1	0.61	NA	NA
	Lite-10	1	1	1	1	1	1	1	1	1	1	1	1	0.64	NA	NA
	DL-1	1	1	1	1	1	1	1	1	1	1	1	1	0.90	0.96	0
	DL-5	1	1	1	1	1	1	1	1	1	1	1	1	0.88	0.97	0
	DL-10	1	1	1	1	1	1	1	1	1	1	1	1	0.88	0.95	0

3.3 Discussions

From our preliminary experiments, we found some interesting problems about OWL ontology systems as well as some issues needed to be further investigated for a complete OWL ontology benchmark.

Native Storage vs DBMS based approaches. OWLIM can be considered as a native ontology repository since it is directly built on the file system. Compared with DBMS based systems (Minerva and DLDB), it greatly reduced the load time. On the other hand, database systems provide many query optimization features, thereby contributing positively to query response time. For OWLIM-like systems, efforts should be made for functionalities such as transactions processing, query optimization, access control and logging/recovery. A typical example is that in query 4, only an exchange of the order of two triples makes OWLIM's response time about 21 times longer (0.6s vs 13s). This suggests that we should leverage DBMS as much as possible. Of course, we also believe that the underlying database more or less affects the performance of ontology systems. For example, DLDB's performance may change when switching the back-end store from Access to SQL server. We are going to investigate such problems.

TBox inference. Considering the modest size of real ontologies (excluding instances), using mature DL reasoners for TBox inference could be a good choice. In fact, Minerva and DLDB leverages a DL reasoner (such as Pellet, FaCT) to understand complete class subsumption. These illustrated that the combination of DL reasoners for TBox inference and rules for ABox inference is a promising approach.

Query interface. SPARQL language is increasingly used for RDF graph query by both RDF(S) and OWL ontology systems [12,21-23]. But, OWL is different from RDFS. In OWL, it is possible to define new classes by logical expressions. In this sense, SPARQL is not an appropriate query language for OWL, since it imposes a substantial restriction on the users' query choices. We should pay more attentions to OWL query interface, such as OWL-QL in [24].

Instance generation. Currently, the extended benchmark provides users a number of parameters for scalable instance generation. In [20], Wang et al. proposed a learned probabilistic model to generate instance data set based on representative samples. The objective is to help the users find an ontology system which best fit their data environment. It is worthwhile investigating what kind of parameters should be provided so that the generated instances set can best simulate user's data.

Tunable TBox. Currently, we do not find a class which is practically meaningful and needs cyclic definition in university domain. But in other domains, such as life sciences, realistic ontologies do include cyclic class definition. Therefore, to add cyclic class definition which may not have real meaning in university domain could be valuable. Furthermore, real ontologies vary tremendously in their average use of ontological constructs. To automatically create an ontology that is tunable by complexity (not just at the level of OWL DL and OWL Lite, but also in terms of the quantities of constructs that are used) is also valuable for users to use in their tests. This could be future research work for a compete benchmark.

Update Tests. A practical ontology system should deal with frequent update in an efficient manner. At the same time, system consistency should be guaranteed. We are intended to add update tests in the UOBM.

4 Conclusions

This paper presented important extensions to the Lehigh University Benchmark in terms of inference and scalability testing. The extended benchmark can characterize the performance of OWL ontology systems more completely. Furthermore, a preliminary evaluation for several well-known ontology systems was conducted and some conclusions were drawn for future research. Also, some issues worthy to be further investigated for a complete OWL ontology benchmark were discussed and summarized.

Acknowledgements

The authors would like to thank Jeff Heflin, Yuanbo Guo and Zhengxiang Pan of Lehigh University, Atanas Kiryakov and Damyan Ognyanov of OntoText Lab, Kavitha Srinivas, Achille Fokoue, Aaron Kershenbaum and Edith Schonberg of IBM T.J. Watson Research Center for their constructive suggestions and comments.

References

1. T. Berners-Lee, J. Hendler, O. Lassila, The Semantic WEB, Scientific American, 2001.
2. J. Davies, D. Fensel, F. Harmelen, Eds., Towards the Semantic WEB: Ontology-driven Knowledge Management, England: John Wiley & Sons, Ltd., 2002.
3. P. Hayes, Resource Description Framework (RDF): Semantics, W3C Recommendation, http://www.w3.org/ TR/2004/REC-rdf-mt-20040210/#rdf_entail, 2004.
4. Michael K. Smith, Chris Welty, Deborah L. McGuinness, OWL Web Ontology language Guide, http://www.w3.org/TR/owl-guide/, 2004.
5. Peter F. Patel-Schneider, Patrick Hayes, Ian Horrocks, OWL Web Ontology Language Semantics and Abstract Syntax, http://www.w3.org/TR/owl-semantics/, 2004.
6. Michael J. Carey, David J. DeWitt, Jeffrey F. Naughtor, "The 007 Benchmark", Proc. of ACM international conference on Management of data, Volume22, Issue 2, 1993.
7. TPC Database Benchmark, http://www.tpc.org/, 2004.
8. Z. Pan, and J. Heflin, "DLDB: Extending Relational Databases to Support Semantic Web Queries", Proc. of Workshop on Practical and Scaleable Semantic Web Systms, pp. 109-113, 2003.
9. Y. Guo, Z. Pan, and J. Heflin, "An Evaluation of Knowledge Base Systems for Large OWL Datasets", Proc. of Third International Semantic Web Conference, pp. 274-288, 2004.

10. The Lehigh University Benchmark, http://swat.cse.lehigh.edu/projects/LUBMm/index.htm, 2004.
11. Pellet OWL Reasoner, http://www.mindswap.org/2003/pellet/index.shtml, 2004.
12. IBM Integrated Ontology Development Toolkit -- Minerva, http://www.alphaworks.ibm.com/tech/semanticstk, 2005.
13. Kopena, J.B. and Regli, W.C, "DAMLJessKB: A Tool for Reasoning with the Semantic Web", In Proc. of ISWC2003.
14. Broekstra, J. and Kampman, "A. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema", In Proc. of ISWC2002.
15. Sesame, An Open Source RDF Database with Support for RDF Schema Inferencing and Querying, http://www.openrdf.org/, 2002.
16. C. Tempich and R. Volz, "Towards a benchmark for Semantic Web reasoners–an analysis of the DAML ontology library", In Workshop on Evaluation on Ontology-based Tools, ISWC2003.
17. R. Volz, Web Ontology Reasoning with Logic Databases. PhD thesis, AIFB, Karlsruhe, 2004.
18. Q. Elhaik, M.C. Rousset, and B. Ycart, "Generating Random Benchmarks for Description Logics". In Proc. of DL' 98.
19. I. Horrocks, and P. Patel-Schneider, "DL Systems Comparison", In Proc. of DL' 98.
20. S. Wang, Y. Guo, A. Qasem, and J. Heflin, "Rapid Benchmarking for Semantic Web KnowledgeBase Systems", Lehigh University Technical Report LU-CSE-05-026, 2005.
21. OWLIM, OWL Semantic Repository, http://www.ontotext.com/owlim/. 2005.
22. KAON2, http://kaon2.semanticweb.org/.
23. Jena2, http://www.hpl.hp.com/semweb/jena.htm.
24. R. Fikes, P. Hayes, I. Horrocks, "OWL-QL - a language for deductive query answering on the Semantic Web", J. of Web Semantics (2004)

Appendix

Format: [Query No.] Query in form of SPARQL
Description explains the meaning of queries and major inference rules involved.

[Query 1] SELECT DISTINCT ?x
WHERE { ?x rdf:type benchmark:UndergraduateStudent . ?x benchmark:takesCourse http://www.Department0.University0.edu/Course0}
Description: All undergraduate students who take course http://www.Department0.University0.edu/Course0. It only needs simple conjunction

[Query 2] SELECT DISTINCT ?x
WHERE { ?x rdf:type benchmark:Employee }
Description: Find out all employees
Domain(worksFor, Employee), <a worksFor b> → <a rdf:type Employee>
Domain(worksFor,Employee), researchAssistant ⊑ ∃worksFor.ResearchGroup→researchAssistant⊑ Employee

[Query 3] SELECT DISTINCT ?x
WHERE {?x rdf:type benchmark:Student . ?x benchmark:isMemberOf http://www.Department0.University0.edu }
Description: Find out all students of http://www.Department0.University0.edu
Range(takeCourse,Student) , GraduateStudent ⊑ ⩾1 takeCourse → GraduateStudent ⊑ Student

[Query 4] SELECT DISTINCT ?x
WHERE { ?x rdf:type benchmark:Publication . ?x benchmark:publicationAuthor ?y .
?y rdf:type benchmark:Faculty . ?y benchmark:isMemberOf http://www.Department0.University0.edu }
Description: All the publications by faculty of http://www.Department0.University0.edu
SubClass: Faculty = FullProfessor ⊔ AssociateProfessor ⊔…⊔ClericStaff, Publication=Article ⊔ …⊔ Journal

[Query 5] SELECT DISTINCT ?x
WHERE { ?x rdf:type benchmark:ResearchGroup . ?x benchmark:subOrganizationOf http://www.University0.edu }
Description: All research groups of http://www.University0.edu
Transitive(subOrganizationOf), <a subOrganizationOf b>, <b subOrganizationOf http://www.University0.edu> → <a subOrganizationOf http://www.University0.edu>

[Query 6] SELECT DISTINCT ?x
WHERE { ?x rdf:type benchmark:Person . http://www.University0.edu benchmark:hasAlumnus ?x }
Description: All alumni of http://www.University0.edu
Inverse(hasAlumni, hasDegreeFrom), <a hasDegreeFrom b> → <b hasAlumnus a>

[Query 7] SELECT DISTINCT ?x
WHERE {?x rdf:type benchmark:Person . ?x benchmark:hasSameHomeTownWith http://www.Department0.University0.edu/FullProfessor0}
Description: Those who has same home town with http://www.Department0.University0.edu/FullProfessor0
Transitive(hasSameHomeTownWith), Symmetric(hasSameHomeTownWith), <a hasSameHomeTownWIth b>, <c hasSameHomeTownWIth b> → < a hasSameHomeTownWith c>

[Query 8] SELECT DISTINCT ?x
WHERE {?x rdf:type benchmark:SportsLover . http://www.Department0.University0.edu benchmark:hasMember ?x}
Description: All sports lovers of http://www.Department0.University0.edu
<x like y>, <y rdf:type Sports>, SportLover⊑∃like.Sports → <x rdf:type SportLover>
subProperty(isCrazyAbout, like), SportFan⊑∃isCrazyAbout.Sports→ SportFan ⊑ SportLover

[Query 9] SELECT DISTINCT ?x
WHERE { ?x rdf:type benchmark:GraduateCourse . ?x benchmark:isTaughtBy ?y .
?y benchmark:isMemberOf ?z .?z benchmark:subOrganizationOf http://www.University0.edu }
Description: All Graduate Courses of http://www.University0.edu
GraduateStudent≡∀takesCourse.GraduateCourse, <a rdf:type GraduateStudent>, <a takesCourse b> → <b rdf:type GraduateCourse>

[Query 10] SELECT DISTINCT ?x
WHERE { ?x benchmark:isFriendOf http://www.Department0.University0.edu/FullProfessor0}
Description: All friends of http://www.Department0.University0.edu/FullProfessor0
Symmetric(isFriendOf), <a isFriendOf b> → <b isFriendOf a>

[Query 11] SELECT DISTINCT ?x
WHERE { ?x rdf:type benchmark:Person . ?x benchmark:like ?y . ?z rdf:type benchmark:Chair . ?z benchmark:isHeadOf http://www.Department0.University0.edu . ?z benchmark:like ?y}
Description: All people who has same interest with the chair of http://www.Department0.University0.edu
FunctionalProperty(isHeadOf), <a isHeadof b>, <c isHeadOf b) → <a sameAs c> // there are some same individuals of chair0

[Query 12] SELECT DISTINCT ?x
WHERE {?x rdf:type benchmark:Student . ?x benchmark:takesCourse ?y
.?y benchmark:isTaughtBy http://www.Department0.University0.edu/FullProfessor0 }
Description All students who take course taught by http://www.Department0.University0.edu/FullProfessor0
GraduateStudent ≡ ∀takesCourse.GraduateCourse ⊓ ⩾1.takesCourse, Domain(takesCourse, Student) → Student ⊒ GraduateStudent

[Query 13] SELECT DISTINCT ?x
WHERE { ?x rdf:type benchmark:PeopleWithHobby . ?x benchmark:isMemberOf http://www.Department0.University0.edu}
Description All people who has some kind of hobbies in http://www.Department0.University0.edu
Lite Cardinality: PeopleWithHobby(⩾1like) ⊒ SportLover, <a like b> → <a rdf:type PeopleWithHobby>

Queries Only for DL

[Query 8]: This query is the same as query 8 in lite, but in context of OWL DL, it will involve more inference rules
Description: Inference rules: SwimmingLover⊑∃like.{Swimming}→ SwimmingLover ⊑ SportsLover …

[Query 14] SELECT DISTINCT ?x
WHERE { ?x rdf:type benchmark:Woman . ?x rdf:type benchmark:Student . ?x benchmark:isMemberOf ?y . ?y benchmark:subOrganizationOf http://www.University0.edu }
Description: All woman students of http://www. University0.edu
<a,isStudentof b>, <b rdf:type WomanCollege>, WomanCollege ⊑∀hasStudent.(¬Man), disjoint(Man, Woman), Man⊔Woman ≡ Person → <a rdf:type Woman>

[Query 15] SELECT DISTINCT ?x
WHERE {?x rdf:type benchmark:PeopleWithManyHobbies . ?x benchmark:isMemberOf http://www.Department0.University0.edu }
Description: All people who has many hobbies in http://www. Department0.University0.edu
PeopleWithManyHobbies⊑⩾3like, <a like b1> … <a like bn>, all different(b1,b2…bn) → <a rdf:type PeopleWithManyHobbies> // n⩾ 3