

# Comparison of Ontology Reasoning Systems Using Custom Rules

Hui Shi, Kurt Maly, Steven Zeil, and Mohammad Zubair

Department of Computer Science  
Old Dominion University, Norfolk, VA 23529, USA  
001-(757) 683-6001

{hshi,maly,zeil,zubair}@cs.odu.edu

## ABSTRACT

In the semantic web, content is tagged with “meaning” or “semantics” that allows for machine processing when implementing systems that search the web. General question/answer systems that are built on top of reasoning and inference face a number of difficult issues. In this paper we analyze scalability issues in the context of a question/answer system (called ScienceWeb) in the domain of a knowledge base of science information that has been harvested from the web. In ScienceWeb we will be able to answer questions that contain qualitative descriptors such as “groundbreaking”, “top researcher”, and “tenurable at university x”. ScienceWeb is being built using ontologies, reasoning systems and custom based rules for the reasoning system. In this paper we address the scalability issue for a variety of supporting systems for ontologies and reasoning. In particular, we study the impact of using custom inference rules that are needed when processing queries in ScienceWeb.

## Categories and Subject Descriptors

C.4 [Computer Systems Organization]: PERFORMANCE OF SYSTEMS – Measurement techniques

## General Terms

Measurement, Performance, Experiment

## Keywords

Semantic Web; Ontology; Ontology Reasoning System; Custom Rules

## 1. INTRODUCTION

The Internet is entering the Web 3.0 phase, where semantic applications can be realized in a collaborative environment. A variety of structured and semi-structured information is available on Internet which can be mined, organized, and queried in a collaborative environment. We are developing one such system, ScienceWeb, with a focus on scientific research. The objective of

the system is to provide answers to qualitative queries that represent the evolving consensus of the community of researchers. The system allows qualitative queries such as: “Who are the groundbreaking researchers in data mining?” or “Is my record tenurable at my university?” We believe qualitative measures are likely to be accepted as meaningful if they represent a consensus of the target community (for example data mining community for the groundbreaking question). Also, with time the qualitative descriptors such as “groundbreaking” can evolve. This requirement raises a number of challenging questions. One question, which is the focus of this paper, is what is the best reasoning system available that work with an evolving knowledge base in a collaborative environment. Supporting a collaborative environment where users are customizing qualitative queries requires a fast response from the reasoning system. This becomes a challenging task for a reasoning system when dealing with custom rules required for qualitative queries. Additionally, we want the reasoning system to work with ontologies, as we plan to use a web ontology language (OWL) to represent the knowledge base in ScienceWeb. In this paper, we evaluate the suitability of several available reasoning systems for ScienceWeb. More specifically, we evaluate the performance of ontology reasoning systems that support custom rules.

Recently a number of semantic applications have been proposed and developed, for example AquaLog[1], QuestIO[2], and QUICK[3]. They are all question-answering systems which take natural language queries as input, then returns answers after processing the queries against the semantic data. Ontology-based semantic markup is widely supported in the Semantic Web, which makes qualitative questions and precise personalized answers possible. Elements of the Semantic Web [4] are the Resource Description Framework (RDF), RDF Schema(RDFS), and the Web Ontology Language (OWL) that are used to describe concepts and relationships in a specific knowledge domain, SPARQL, a W3C standard query language for RDF [5], and SWRL [6], a web rule language combining OWL and RuleML to express custom rules. Jena[7] supports custom rules in its proprietary format “Jena Rules”. Pellet[8] and KAON2[9] both support SWRL. Oracle 11g [10] provides native inference in the database for user-defined rules. OWLIM[11] could specify custom rule-sets(rules + axiomatic triples).

A number of studies have been done on the performance of these OWL-based reasoning systems. These studies address the scalability of the above mentioned reasoning systems with regard to the complexity of the ontology and the number of triples in the ontology. Queries tested rely on the native rule system a particular

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WIMS'11, May 25-27, 2011 Sogndal, Norway

Copyright © 2011 ACM 978-1-4503-0148-0/11/05... \$10.00

reasoner supports. We are not aware of any study that includes custom defined rules. Customized rules are needed when qualitative descriptors involve transitive reasoning as in “What is the PHD advisor genealogy of professor x?” or complex queries that involve algebraic computation across a populations such as in “groundbreaking”.

In this paper, we evaluate the performance of Jena, Pellet, KAON2, Oracle 11g and OWLIM using representative custom rules, including transitive and recursive rules, on top of our ontology and LUBM [12]. In order to support custom rules, they do need to combine OWL inference and custom rule inference. These systems support custom rules in different formats and degrees. We will compare how the size of the Abox and the distribution of the Abox affect their inference performance for different samples of custom rules. We also compare query performance and query cache effects.

The remainder of this paper is organized as follows: In Section 2 we discuss related work on existing benchmarks and give a general description of OWL reasoners and the format of the custom rules that they support. In Section 3, we discuss an experimental design to compare these ontology reasoning systems with custom rules. In Section 4, results are presented, and we compare these systems based on the experiments on two ontologies. Conclusions are given In Section 5.

## 2. RELATED WORK

### 2.1 Existing Benchmarks

There are several ontology toolkits, such as Jena [7], Pellet [8], KAON2 [9], Oracle 11g [10] and OWLIM [11] for reasoning, querying, and storing ontologies. Researchers have evaluated these systems in order to help people make their decision for choosing an appropriate toolkit for a given semantic application task with specific requirements on performance, complexity and expressiveness and scalability.

Lee et al. [13] evaluate four of the most popular ontology reasoning systems using query sequences. The test data and queries are all from LUBM (Leigh University Benchmark) [12]. They classify queries by selectivity and complexity and evaluate the response time of each individual query and elapsed time of each query sequence. KAON2, Pellet and Racer have been compared in [14] for six different kinds of ontologies. Their results indicate that KAON has good performance on knowledge bases with large Aboxes, while other systems provide superior performance on knowledge bases with large and complex Tboxes. Gardiner et al. [15] aim to build an automated benchmarking system, which allows the automatic quantitative testing and comparison of DL reasoners on real-world examples. Jürgen Bock et al. [16] classify the existing ontologies on the web into four clusters by expressivity. They chose four representative ontologies for each cluster to analyze the capability of the OWL reasoners. They conclude that RacerPro performs better in settings with high complexity and small Aboxes. OWLIM is the recommendation for low complexity settings while KAON2 is better for all other cases [16]. LUBM [12] has been widely used in many evaluations of ontology reasoning systems. It is an ontology that focuses on a university domain. LUBM provides varying sizes of OWL data and fourteen representative queries. LUBM only covers a subset

of OWL Lite. The University Ontology Benchmark (UOBM)[17] is an extension of LUBM. It is OWL DL complete. In [17] Three ontology reasoning systems have been evaluated based on UOBM.

Weithöner et al. [18] point out the limitations of the existing benchmarks and gives a set of general benchmarking requirements that will be helpful for the future OWL reasoner benchmarking suites. Rohloff et al.[19] compare the performance of triple stores based on LUBM.

### 2.2 Ontology Reasoning Systems Supporting Custom Rules

#### 2.2.1 Jena

Jena is a Java framework for Semantic Web applications that includes an API for RDF. It supports in-memory and persistent storage, a SPARQL query engine and a rule-based inference engine for RDFS and OWL. The rule-based reasoner implements both the RDFS and OWL reasoners. It provides forward and backward chaining and a hybrid execution model. It also provides ways to combine inferencing for RDFS/OWL with inferencing over custom rules.[7]

#### 2.2.2 Pellet

Pellet is a free open-source Java-based reasoner supporting reasoning with the full expressivity of OWL-DL (SHOIN(D) in Description Logic terminology). It is a great choice when sound and complete OWL DL reasoning is essential. It provides inference services such as consistency checking, concept satisfiability, classification and realization. Pellet also includes an optimized query engine capable of answering Abox queries. DL-safe custom rules can be encoded in SWRL.[8]

#### 2.2.3 KAON2

KAON2 is a free (free for non-commercial usage) Java reasoner including API for programmatic management of OWL-DL, SWRL, and F-Logic ontologies. It supports all of OWL Lite and all features of OWL-DL apart from nominals. It also has an implementation for answering conjunctive queries formulated using SPARQL. Much, but not all, of the SPARQL specification is supported. SWRL is the way to support custom rules in KAON2.[9]

#### 2.2.4 Oracle 11g

As an RDF store, Oracle 11g provides APIs that supports the Semantic Web. Oracle 11g provides full support for native inference in the database for RDFS, RDFS++, OWLPRIME, OWL2RL, etc. Custom rules are called user-defined rules in Oracle 11g. It uses forward chaining to do the inference. [10] In Oracle the custom rules are saved as records in tables.

#### 2.2.5 OWLIM

OWLIM is both a scalable semantic repository and reasoner that supports the semantics of RDFS, OWL Horst and OWL 2 RL. SwiftOWLIM is a “standard” edition of OWLIM for medium data volumes. It is the edition in which all the reasoning and query are performed in memory.[11] Custom rules are defined via rules and axiomatic triples.

We summarize the different ontology based reasoning systems along with the features they support in Table 1.

**Table 1 General comparison among ontology reasoning systems**

|            | Supports RDF(S)? | Supports OWL? | Rule Language | Supports SPARQL Queries? | Persistent Repository (P) or In-Memory (M) |
|------------|------------------|---------------|---------------|--------------------------|--------------------------------------------|
| Jena       | yes              | yes           | Jena Rules    | yes                      | M                                          |
| Pellet     | yes              | yes           | SWRL          | no                       | M                                          |
| KAON2      | yes              | yes           | SWRL          | yes                      | M                                          |
| Oracle 11g | yes              | yes           | Owl Prime     | no                       | P                                          |
| OWLIM      | yes              | yes           | Owl Horst     | yes                      | P                                          |

### 3. COMPARISON OF ONTOLOGY REASONING SYSTEMS: EXPERIMENTAL DESIGN

The basic question addressed in this study is one of scalability. How effective are the various existing reasoning systems in answering queries when customized rules are added to native logic? Can these systems handle millions of triples in real time? To be able to answer this question, in this section we shall discuss the:

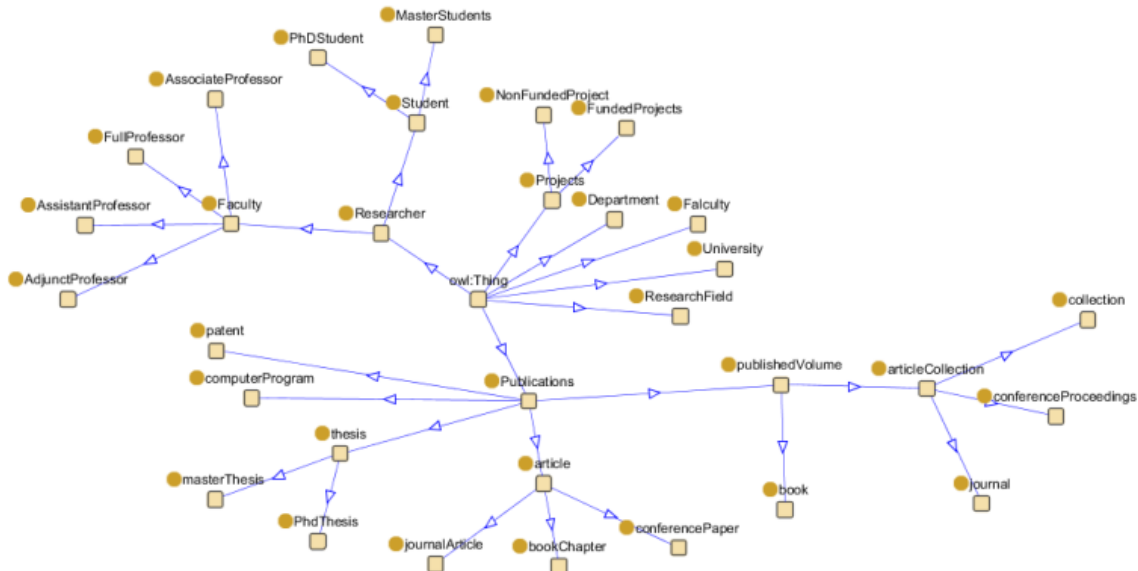
- data (ontology classes/relations and instance data) that will be involved in querying the ontology reasoning systems
- the custom rules that are going to be used
- environment and metrics to evaluate a system
- evaluation procedure

#### 3.1 Ontology data for the experiments

As described in section 2, there have been a number of studies for reasoning systems using only their native logic. To provide credibility for our context, we shall use benchmark data from these studies and replicate their results with native logic and then

extend them by adding customized rules. We will use LUBM for the purpose of comparing our results with earlier studies. The second set of data will emulate a future ScienceWeb. Since ScienceWeb at this time is only a Gedanken experiment, we need to use artificial data for that purpose. Figure 1 shows the limited ontology class tree we deem sufficient to explore the scalability issues.

Both the LUBM and the ScienceWeb ontologies are about concepts and relationships in a research community. For instance, concepts such as *Faculty*, *Publication*, and *Organization* are included in both ontologies, as are properties such as *advisor*, *publicationAuthor*, and *worksFor*. All the concepts of LUBM can be found in the ScienceWeb ontology, albeit the exact name for classes and properties may not be same. ScienceWeb will provide more detail for some classes. For example, the ScienceWeb ontology has a smaller granularity when it describes the classification and properties of *Publication*. The ScienceWeb ontology shown in Figure 1 represents only a small subset of the one to be used for ScienceWeb. It was derived to be a minimal subset that is sufficient to answer a few select qualitative queries. The queries were selected to test the full capabilities of a reasoning system and to necessitate the addition of customized rules.



**Figure 1 Class tree of research community ontology.**

Both LUBM and ScienceWeb provide both an ontology and sample data (or data generators). LUBM starts with a university and generates faculty members for that university. The advisor of a student must be a faculty in the same university. The coauthors of a paper must be in the same university. In ScienceWeb we generate data that reflect a more realistic situation where faculty can have advisors at different universities and, co-authors can be at different universities. In another paper (in preparation) we describe a tool called UnivGenerator that will generate a specified number of ontology triples within specific constraints. For instance, the user can specify that the number of publications of an associate professor ranges within 10-15 publications and the number of co-authors ranges from 0-4. The generator will ensure that triples are generated within these constraints and will make sure that the relations, e.g. co-author, are properly instantiated. The size range of the datasets in our experiments is listed in Table 2. We generate 7 datasets for LUBM and 9 datasets for ScienceWeb, both ranges from thousand to millions for our experiment.

**Table 2 Size range of datasets (in triples)**

|             | Data set1 | Data set2 | Data set3 | Data set4 | Data set5 |
|-------------|-----------|-----------|-----------|-----------|-----------|
| Science Web | 3511      | 6728      | 13244     | 166163    | 332248    |
| LUBM        | 8814      | 15438     | 34845     | 100838    | 624827    |
|             | Data set6 | Data set7 | Data set8 | Data set9 |           |
| Science Web | 1327573   | 2656491   | 3653071   | 3983538   |           |
| LUBM        | 1272870   | 2522900   |           |           |           |

### 3.2 Custom Rule Sets and Queries

We now present the 5 rule sets and 3 corresponding queries that we shall use in the experiments. Rule sets were defined to test basic reasoning to allow for validation, such as co-authors having to be different, and to allow for transitivity and recursion. Rule sets 1 and 2 are for the co-authorship relation, rule set three is used in queries for the genealogy of PhD advisors (transitive) and rule set 4 is to enable queries for “good” advisors. Rule set 5 is a combination of the first 4 sets.

#### Rule set 1: Co-author

authorOf(?x, ?p)  $\wedge$  authorOf(?y, ?p)  
 $\Rightarrow$  coAuthor(?x, ?y)

#### Rule set 2: validated Co-author

authorOf(?x, ?p)  $\wedge$  authorOf(?y, ?p)  $\wedge$  notEqual(?x, ?y)  
 $\Rightarrow$  coAuthor(?x, ?y)

#### Rule set 3: Research ancestor (transitive)

advisorOf(?x, ?y)  $\Rightarrow$  researchAncestor(?x, ?y)  
researchAncestor(?x, ?y)  $\wedge$  researchAncestor(?y, ?z)  
 $\Rightarrow$  researchAncestor(?x, ?z)

#### Rule set 4: Distinguished advisor (recursive)

advisorOf(?x, ?y)  $\wedge$  advisorOf(?x, ?z)  $\wedge$  notEqual(?y, ?z)  
 $\wedge$  worksFor(?x, ?u)  
 $\Rightarrow$  distinguishAdvisor(?x, ?u)

advisorOf(?x, ?y)  $\wedge$  distinguishAdvisor(?y, ?u)  $\wedge$  worksFor(?x, ?d)  
 $\Rightarrow$  distinguishAdvisor(?x, ?d)

#### Rule set 5: combination of above 4 rule sets.

In the Jena rules language, these rules are encoded as:

@include <OWL>.

```
[rule1: (?x uni:authorOf ?p) (?y uni:authorOf ?p) notEqual(?x, ?y)
-> (?x uni:coAuthor ?y)]
[rule2: (?x uni:advisorOf ?y) -> (?x uni:researchAncestor ?y)]
[rule3: (?x uni:researchAncestor ?y) (?y uni:researchAncestor ?z)
-> (?x uni:researchAncestor ?z)]
[rule4: (?x uni:advisorOf ?y) (?x uni:advisorOf ?z notEqual(?y, ?z)
(?x uni:worksFor ?u) -> (?x uni:distinguishAdvisor ?u)]
[rule5: (?x uni:advisorOf ?y) (?y uni:distinguishAdvisor ?u) (?x
uni:worksFor ?d) -> (?x uni:distinguishAdvisor ?d)]
```

In SWRL these rules are less compact. Rule 1 would be encoded as

```
<swrl:Variable rdf:about="#x"/>
<swrl:Variable rdf:about="#y"/>
<swrl:Variable rdf:about="#p"/>
<swrl:Imp rdf:about="rule1">
  <swrl:head rdf:parseType="Collection">
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="#coAuthor"/>
      <swrl:argument1 rdf:resource="#x"/>
      <swrl:argument2 rdf:resource="#y"/>
    </swrl:IndividualPropertyAtom>
  </swrl:head>
  <swrl:body rdf:parseType="Collection">
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="#authorOf"/>
      <swrl:argument1 rdf:resource="#x"/>
      <swrl:argument2 rdf:resource="#p"/>
    </swrl:IndividualPropertyAtom>
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="#authorOf"/>
      <swrl:argument1 rdf:resource="#y"/>
      <swrl:argument2 rdf:resource="#p"/>
    </swrl:IndividualPropertyAtom>
    <swrl:DifferentIndividualsAtom>
      <swrl:argument1 rdf:resource="#x"/>
      <swrl:argument2 rdf:resource="#y"/>
    </swrl:DifferentIndividualsAtom>
  </swrl:body>
</swrl:Imp>
```

In Owlrim, rule1 would be encoded as:

```
Id: rule1
x <uni:authorOf> p [Constraint x != y]
y <uni:authorOf> p [Constraint y != x]
-----
x <uni:coAuthor> y [Constraint x != y]
```

We have composed 3 queries to use in these tests, expressed in SPARQL notation:

#### Query 1: Co-author

```
PREFIX uni:<http://www.owl-
ontologies.com/OntologyUniversityResearchModel.owl#>
SELECT ?x ?y
WHERE { ?x uni:coAuthor ?y. ?x uni:hasName
  \FullProfessor0_d0_u0\ }
```

### Query 2: Research ancestor

```
PREFIX uni:<http://www.owl-  
ontologies.com/OntologyUniversityResearchModel.owl#>  
SELECT ?x ?y  
WHERE {?x uni:researchAncestor ?y. ?x uni:hasName  
      \"FullProfessor0_d0_u0\" };
```

### Query 3: Distinguished advisor

```
PREFIX uni:<http://www.owl-  
ontologies.com/OntologyUniversityResearchModel.owl#>  
SELECT ?x ?y  
WHERE {?x uni:distinguishAdvisor ?y. ?y uni:hasTitle  
      \"department0u0\" };
```

Queries are used with the rules sets that define the properties employed in the queries. Rule sets 1 and 2 are tested with query 1. Rule set 2 is tested with query 2. Rule sets 3 and 4 are tested with queries 2 and 3. Rule set 5 is tested with all queries.

## 3.3 Experimental Environment and Metrics

The latest version owl reasoning systems supporting custom rules have been chosen for the evaluation: Jena (2.6.2, 2009-10-22 release), KAON2 (2008-06-29 release), Pellet (2.2.0, 2010-07-06 release), OWLIM (3.3, 2010-07-22 release), and Oracle 11g R2 (11.2.0.1.0, 2009-09-01 release). As we wanted to insure that we were obtaining results that were commensurate with the earlier benchmark studies, we have taken the two main metrics from [20]. These are:

- Setup time: This stage includes loading and preprocessing time before any query can be made. This includes loading the ontology and instance data into the reasoning system, and any parsing, inference that needs to be done.
- Query processing time: This stage starts with parsing and executing the query and ends when all the results have been saved in the result set. It includes the time of traversing the result set sequentially. For some systems (Jena, Pellet, Kaon) it might include some preprocessing work.

All the tests were performed on a PC with a 2.40 GHz Intel Xeon processor and 16 G memory, running Windows Server 2008 R2 Enterprise. Sun Java 1.6.0 was used for Java-Based tools. The maximum heap size was set to 800M. We defined one hour as the time-out period.

## 3.4 Evaluation Procedure

Our goal is to evaluate the performance of different ontology reasoning systems in terms of reasoning and querying time using custom rules.

We are interested in two aspects of scalability. One aspect is simply the size of data. That is, we are interested in the performance of a system as the number of triplet's changes from small toy size to realistic sizes of millions. A second aspect of scalability we are interested is the complexity of reasoning. That is, we are interested what some of the limits are in the type of questions ScienceWeb users can ask. To that end, we will perform the experiments with the transitive rules where we have different limits on the transitive chain.

Finally we are interested in the impact of using realistic models of the instance space (ScienceWeb) versus a simpler model (LUBM).

## 4. RESULTS AND DISCUSSION

### 4.1 Comparison on the Full Rule Sets

#### 4.1.1 Setup Time

We begin by comparing setup times for the five systems under investigation. Each system is examined using data sets derived from the two ontologies discussed in Section 3.1. The generated data sets contained thousands to millions of triples of information. The setup time can be sensitive to the rules sets used for inferencing about the data because rules can entail filters (e.g., “not equal”), transitivity, or recursion that will consequently involve different Abox reasoning with different sizes and distributions of Aboxes. Therefore each system-ontology pair was examined using each of the 5 rules sets presented in Section 3.2.

Figure 2 shows the setup time for LUBM ontology. Figure 3 shows the setup time for ScienceWeb ontology. For some systems, data points are missing for the larger size of Abox. This is because Jena, Pellet and KAON2 load the entire data set into memory when performing inferencing. For these systems, therefore, the maximum accepted size of the Abox depends on the size of memory. In our environment, Jena and Pellet could only provide inferencing for small Aboxes (less than 600000 triples). KAON2 was able to return answers for Aboxes of up to 2 million triples. OWLIM and Oracle have better scalability because they are better able to exploit external memory for inference. They both are able to handle the largest data set posed, which is nearly 4 million triples. As the dataset grows, it turns out Oracle has the best scalability, which is not shown in the figures.

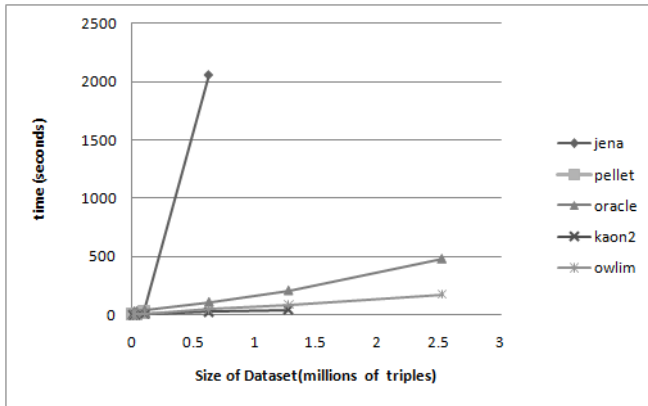


Figure 2(a) Setup time of rule set 1 for LUBM dataset

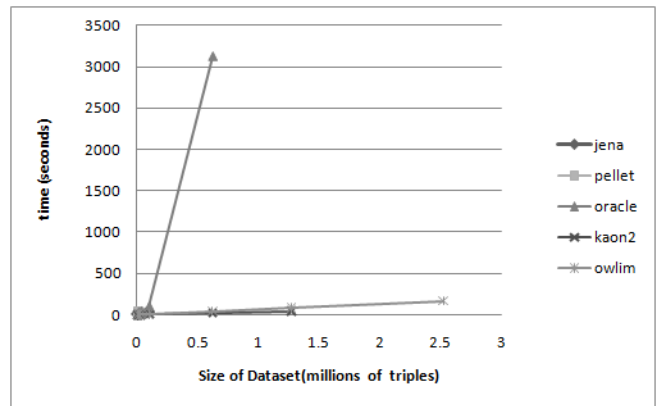


Figure 2(b) Setup time of rule set 2 for LUBM dataset

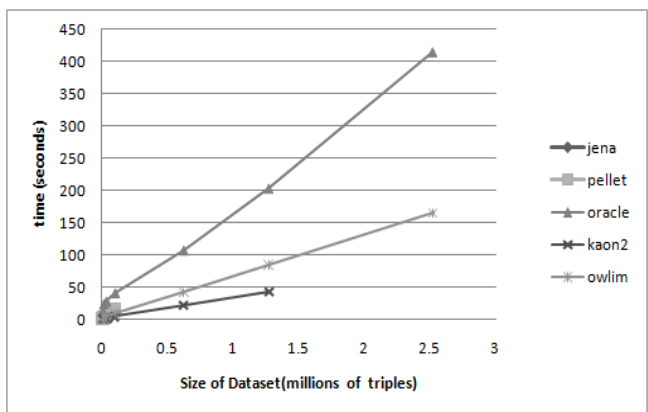


Figure 2(c) Setup time of rule set 3 for LUBM dataset

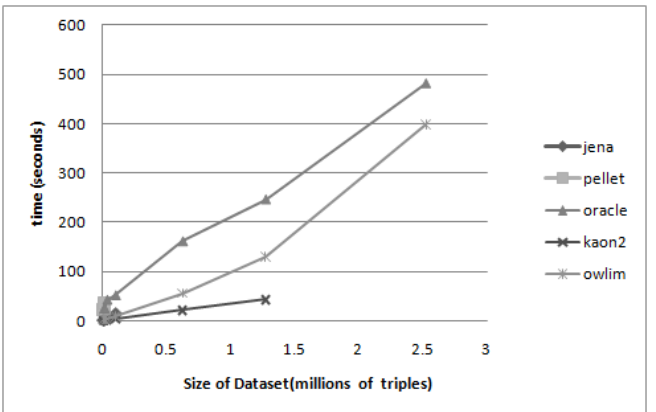


Figure 2(d) Setup time of rule set 4 for LUBM dataset

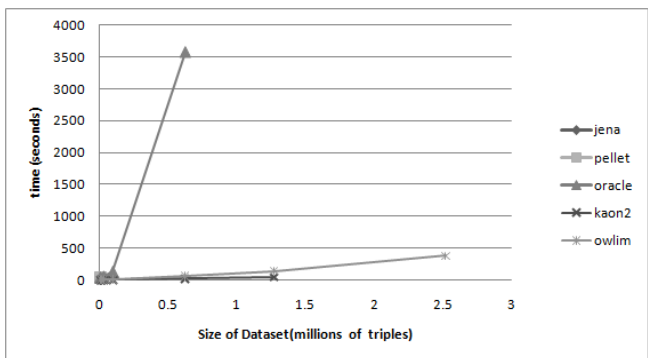


Figure 2(e) Setup time of rule set 4 for LUBM dataset

Figure 2 Setup times of rule sets for LUBM dataset

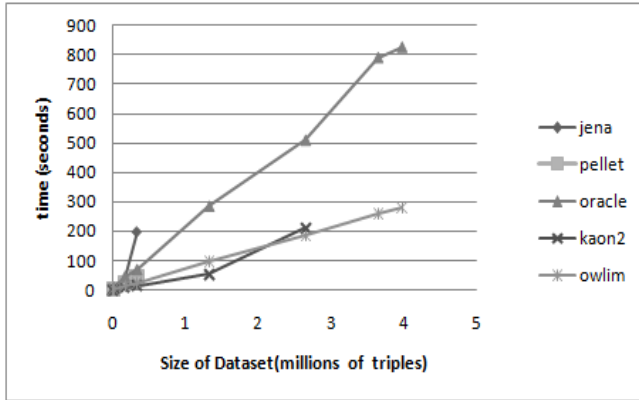


Figure 3(a) Setup time of rule set 1 for ScienceWeb dataset

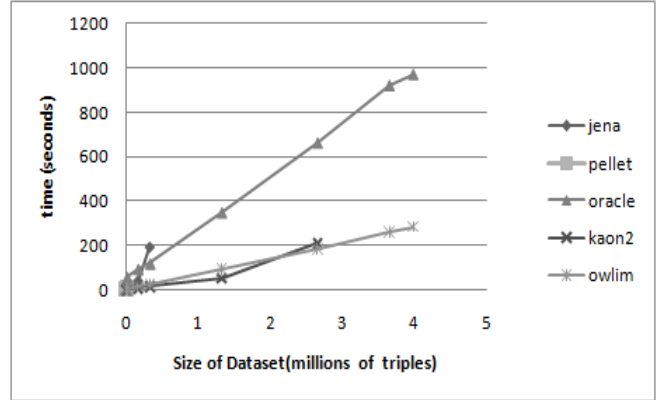


Figure 3(b) Setup time of rule set 2 for Science Web dataset

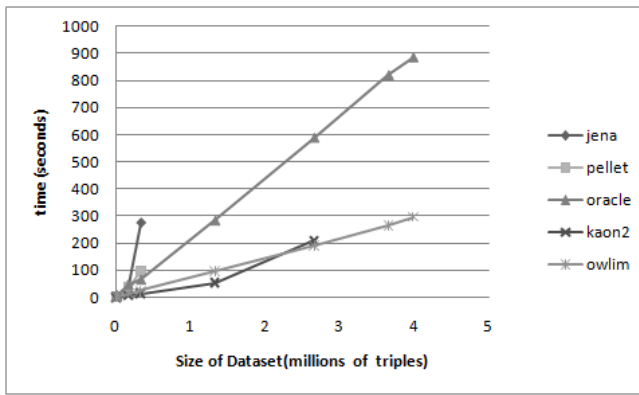


Figure 3(c) Set-up time of rule set 3 for ScienceWeb dataset

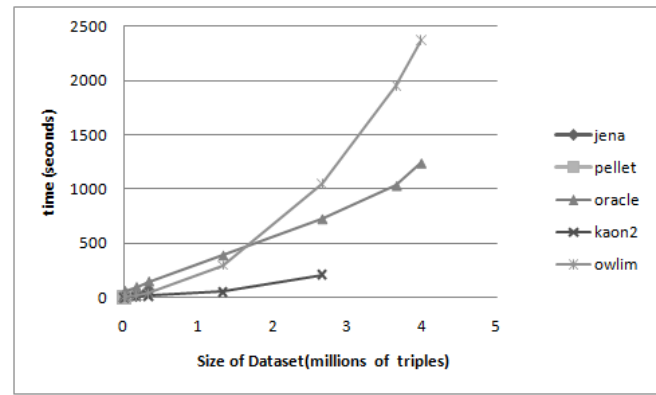


Figure 3(d) Setup time of rule set 4 for ScienceWeb dataset

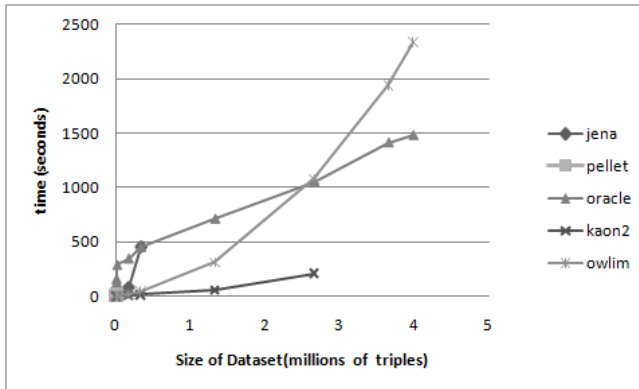


Figure 3(e) Setup time of rule set 5 for ScienceWeb dataset

Figure 3 Setup times of rule sets for ScienceWeb dataset

For smaller Aboxes (less than 2 million triples), KAON2 usually has the best performance on setup time. But as the size of the Abox grows past that point, only Oracle 11g and OWLIM are able to finish the setup before time-out.

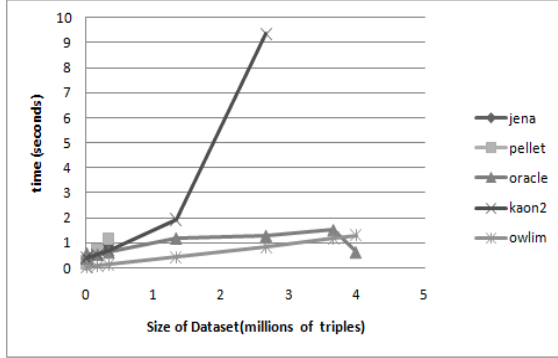
The performances of these systems vary considerably, especially when the size of dataset grows. OWLIM performs better than Oracle 11g on rule sets 1, 2 and 3, as shown in Figure 2. But, as

shown in Figure3(d) and Figure3(e), Oracle has better performance, which means OWLIM is not good at rule set 4 in ScienceWeb dataset. This appears to be because the ScienceWeb dataset involves more triples in the Abox inference for rule set 4 than does the LUBM dataset. Thus, when large Abox inferencing is involved in set-up, Oracle 11g performs better than OWLIM. Oracle 11g needs to set a “filter” when inserting rule set 2 into the

database to implement “notEqual” function in this rule set. As the Abox grows, more data has to be filtered while creating the entailment. This filter significantly slows the performance. For LUBM, Oracle 11g is not able to finish the setup for a one million triple Abox in one hour while OWLIM could finish in 100 seconds. As shown in Figure 2(b) and Figure 2(e), the data points are missing for larger size of datasets.

#### 4.1.2 Query Processing Time

Based on our results, OWLIM has the best query performance in most datasets in our experiments, but this superiority is overtaken as the size of dataset grows. Figure 4 shows the time required to process query 1 on the ScienceWeb dataset.

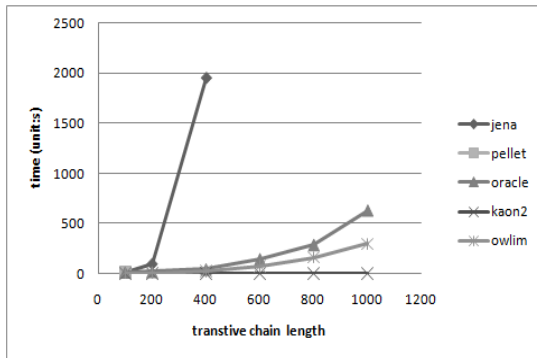


**Figure 4 Query processing time of query 1 for ScienceWeb dataset**

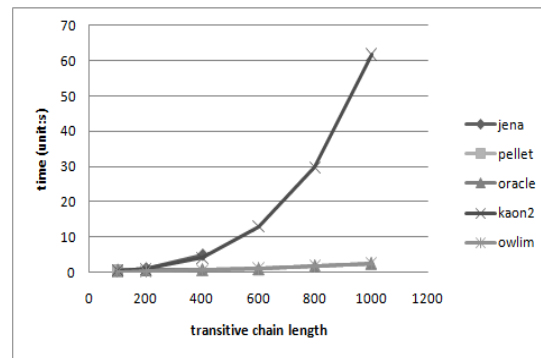
In actual use, one might expect that some inferencing results will be requested repeatedly as component steps in larger queries. If so, then the performance of a reasoning system may be enhanced if such results are cached.

**Table 3 Caching ratios between processing time of single query and average processing time on ScienceWeb ontology for query 1**

|        | 3511 | 6728 | 13244 | 166163 | 332248 | 1327573 | 2656491 | 3653071 | 3983538 |
|--------|------|------|-------|--------|--------|---------|---------|---------|---------|
| Jena   | 6.13 | 5.57 | 5.50  | 2.40   | 1.87   |         |         |         |         |
| Pellet | 6.03 | 5.48 | 4.91  | 1.56   | 1.32   |         |         |         |         |
| Oracle | 5.14 | 2.77 | 2.65  | 5.59   | 3.40   | 3.49    | 3.75    | 3.75    | 8.22    |
| KAON2  | 6.34 | 5.59 | 5.59  | 2.24   | 1.65   | 1.02    | 1.02    |         |         |
| OWLIM  | 1.83 | 1.83 | 1.30  | 1.48   | 1.20   | 1.05    | 1.07    | 1.01    | 1.03    |



**Figure 5 Setup time for transitive rule**



**Figure 6 Query processing time after inference over transitive rule**

Next we therefore record the processing time for single queries and the average processing time for repeated queries. The average processing time is computed by executing the query consecutively 10 times. The average query processing time has been divided by the single query processing time to detect how caching might affect query processing. Table 3 shows the caching ratio between processing time of one single query and average processing time on ScienceWeb ontology for query 1. Based on our results, OWLIM is the only one that does not have a pronounced caching effect. The caching effect of other systems becomes weaker as the size of dataset grows.

## 4.2 Comparison among Systems on Transitive Rule

We anticipate that many ScienceWeb queries will involve transitive or recursive chains of inference. We next examine the behavior of the system on such chains of inference. Because the instance data for the LUBM and ScienceWeb ontologies may not include sufficient long transitive chains, we created a group of separate instance files containing different number of individuals that are related via the transitive rule in rule set 3.

As Figures 5 and 6 show, Pellet only provides the results before time-out when the length of transitive chain is 100. Jena's performance degrades badly when the length is more than 200. Only KAON2, OWLIM and Oracle 11g could complete inference and querying on long transitive chains.

KAON2 has the best performance in terms of setup time, with OWLIM having the second best performance. For query processing time, OWLIM and Oracle have almost same performance in terms of querying time. When the length of transitive chain grows from 500 to 1000, the querying time of KAON2 has a dramatic increase.

## 5. CONCLUSIONS

One of the promises of the evolving Semantic Web is that it will enable systems that can handle qualitative queries such as “good PhD advisors in data mining”. We have explored in this paper the feasibility of developing such a system using ontologies and reasoning system that can handle customized rules such as “good advisor”.

When looking at more realistic models (ScienceWeb) than provided by LUBM serious issues arise when the size approaches million of triplets. For the most part, OWLIM and Oracle offer the best scalability for the kinds of datasets anticipated for ScienceWeb.

This scalability comes in part from heavy front-loading of the inferencing costs by pre-computing the entailed relationships at set-up time. This, in turn, has negative implications for evolving systems. One can tolerate high one-time costs that are needed to set up a reasoning system for a particular ontology and an instance set. If, however, the ontologies or rule sets evolve at a significant rate, then repeated incurrence of these high setup cost is likely to be prohibitively expensive.

The times we obtained for some queries indicate that real-time queries over large triplet spaces will have to be limited in their scope unless one gives up on the answers being returned in real time. The question as to how we can specify what can be asked within a real-time system is an open one.

## 6. REFERENCES

- [1] Lopez, V., M. Pasin, and E. Motta, *Aqualog: An ontology-portable question answering system for the semantic web*. The Semantic Web: Research and Applications, 2005. 3532: p. 546-562.
- [2] Tablan, V., D. Damjanovic, and K. Bontcheva, *A natural language query interface to structured information*, in *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*. 2008: Tenerife, Canary Islands, Spain p. 361-375.
- [3] Zenz, G., et al., *From keywords to semantic queries--Incremental query construction on the semantic web*. Web Semantics: Science, Services and Agents on the World Wide Web, 2009. 7(3): p. 166-176.
- [4] Berners-Lee, T., J. Hendler, and O. Lassila, *The semantic web*. Scientific american, 2001. 284(5): p. 28-37.
- [5] Prud'hommeaux, E. and A. Seaborne, *SPARQL Query Language for RDF (W3C Recommendation 15 January 2008)*. World Wide Web Consortium, 2008.
- [6] Horrocks, I., et al., *SWRL: A semantic web rule language combining OWL and RuleML*. W3C Member submission, 2004. 21.
- [7] Jena team and Hewlett-Packard. *Jena Semantic Web Framework*. 2010 [cited 2010 October,13]; Available from: <http://jena.sourceforge.net/>.
- [8] Clark & Parsia. *Pellet: The Open Source OWL2 Reasoner*. 2010 [cited 2010 October,13]; Available from: <http://clarkparsia.com/pellet/>.
- [9] Information Process Engineering (IPE), Institute of Applied Informatics and Formal Description Methods (AIFB), and I.M.G. (IMG). *KAON2-Ontology Management for the Semantic Web*. 2010 [cited 2010 October,13]; Available from: <http://kaon2.semanticweb.org/>.
- [10] Oracle Corporation. *Oracle Database 11g R2* 2010 [cited 2010 October,13]; Available from: <http://www.oracle.com/database/11g.com>.
- [11] Ontotext. *OWLIM-OWL Semantic Repository*. 2010 [cited 2010 October 13]; Available from: <http://www.ontotext.com/owlim/>.
- [12] Guo, Y., Z. Pan, and J. Heflin, *LUBM: A benchmark for OWL knowledge base systems*. Web Semantics: Science, Services and Agents on the World Wide Web, 2005. 3(2-3): p. 158-182.
- [13] Lee, C., et al., *A comparison of ontology reasoning systems using query sequences*, in *Proceedings of the 2nd international conference on Ubiquitous information management and communication* 2008. p. 543-546.
- [14] Motik, B. and U. Sattler, *A comparison of reasoning techniques for querying large description logic aboxes*. Logic for Programming, Artificial Intelligence, and Reasoning, 2006. 4246: p. 227-241.
- [15] Gardiner, T., I. Horrocks, and D. Tsarkov, *Automated benchmarking of description logic reasoners*, in *Workshop on Description Logics (DL'06)*. 2006: Lake District, UK. p. 167-174.
- [16] Bock, J., et al., *Benchmarking OWL reasoners*, in *Proc. of the ARea2008 Workshop*. 2008: Tenerife, Spain.
- [17] Ma, L., et al., *Towards a complete OWL ontology benchmark*. The Semantic Web: Research and Applications, 2006. 4011: p. 125-139.
- [18] Weithöner, T., et al., *Real-world reasoning with OWL*. The Semantic Web: Research and Applications, 2007. 4519: p. 296-310.
- [19] Rohloff, K., et al., *An evaluation of triple-store technologies for large data stores*, in *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems*. 2007. p. 1105-1114.
- [20] Weithöner, T., et al., *What's wrong with OWL benchmarks*, in *Proceedings of the Second International Workshop on Scalable Semantic Web Knowledge Base Systems*. 2006. p. 101-114.