A Comparison of Ontology Reasoning Systems
Using Query Sequences

Chulki Lee¹  Sungchan Park¹  Dongjoo Lee¹  Jae-won Lee¹  Ok-Ran Jeong²  Sang-goo Lee¹
¹School of Computer Science and Engineering,
Seoul National University
Seoul 151-742, Republic of Korea
{ chulki.lee, baksalchan, therocks, lyonking, sglee}
@europa.snu.ac.kr

ABSTRACT
A number of ontology reasoning systems have been developed for reasoning and querying the semantic web. Since they implement different reasoning algorithms and optimization techniques, they differ in a number of ways. Previous attempts at comparing performance of ontology reasoning systems have mainly considered performances of individual query requests. In this paper, we present the results of testing four of the most popular ontology reasoning systems on query sequences that reflect real world use cases. We believe that using query sequences is a more effective way to evaluate ontology reasoning systems.

Categories and Subject Descriptors
C.4 [Computer Systems Organization]: PERFORMANCE OF SYSTEMS – Measurement techniques

General Terms
Measurement, Performance, Experimentation.

Keywords
Query sequence, Benchmark, Ontology reasoning system

1. INTRODUCTION
Various reasoning systems have been developed for reasoning and querying ontologies. Since they implement different reasoning algorithms and optimization techniques, they differ in a number of ways. There have been several works to compare the performances of ontology reasoning systems [1-6]. All of these except [6] measures the response time of individual queries. Query repetition and query specialization have been tested in [6] to measure caching performance. However sequences made of different queries, which better reflect real world use cases, have been excluded.

We present results of testing a set of ontology reasoning systems considering single queries, repetitions of queries, and query sequences. A query sequence is a sequence of individual query requests mimicking real world behavior. We believe that using query sequences is a more effective way to evaluate ontology reasoning systems.

The remainder of this paper is organized as follows: our benchmark plan is described in Section 2, and results of our experiment are presented in Section 3. We conclude with discussion and future works in Section 4.

2. BENCHMARK WITH QUERY SEQUENCES
2.1 Target Systems
We have selected systems which are frequently referenced in research papers and, at the same time, has at least one published project using the system. For those that were selected, we have used the most recent releases of as follows: Pellet (1.5.1, 2007-10-26 release) [7], KAON2 (2007-10-14 release) [8], Sesame (1.2.7, 2007-06-12 release), Sesame2 (2.0-rc1, 2007-11-12 release) [9], and OWLIM (1.5.1, 2007-10-26 release) [10]. These systems are used in many projects, for example Noesis project [11], SmartWeb project [12], OntoMedia project [13], and SIMDAT project [14].

Sesame1 and Sesame2 provide several repository classes, and thus, results in multiple versions. We have tested mem-rdfs, native-rdf, rdbms-rdfs from Sesame1 and mem-rdf, native-rdf from Sesame2. Mem, native and RDBMS represent the way data is stored; main memory, native file system, and relational DBMS, respectively. The repositories of Sesame1 and Sesame2 support RDF or RDFS inference level. But partial answers will often be acceptable on the Semantic Web.

2.2 Test Data and Queries
LUBM(Leigh University Benchmark) is the ontology benchmark dataset used most widely. LUBM offers an ontology generator which generates ontologies with fixed TBox and size-adjustable ABox. LUBM(N, S) is dataset which contains information of N universities and is generated using a seed value of S. Since our objective is to observe the effects of query sequences, not the effects of larger datasets, we use LUBM(1, 0) in the experiment.

LUBM provides fourteen test queries [2]. We classify queries by selectivity and complexity, adopting the convention from [2] Selectivity is measured as the estimated proportion of the class instances involved in the query that satisfy the query criteria. Selectivity of query is said to be high when the proportion is lower than 10%.
Complexity refers to the number of concepts and properties involved. A query is said to have high complexity if the involved concepts and properties are more than two. If the involved concepts and properties are not more than two, the query is said to have low complexity.

2.3 Test Query Sequences
Ontology reasoning systems typically receive different queries randomly in real use cases. In order to reflect this situation, we define the sequence of input queries as a query sequence. For the purpose of testing the performance of ontology reasoning systems, we define four query sequences (QS1, QS2, QS3, and QS4) each of which consist of 1000 LUBM queries. Table 1 lists the generated query sequences and their characteristics.

Table 1. Compositions of query sequences

<table>
<thead>
<tr>
<th>Name</th>
<th>Complexity</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>QS1</td>
<td>71% low</td>
<td>84% low</td>
</tr>
<tr>
<td>QS2</td>
<td>85% low</td>
<td>71% high</td>
</tr>
<tr>
<td>QS3</td>
<td>79% high</td>
<td>73% low</td>
</tr>
<tr>
<td>QS4</td>
<td>71% high</td>
<td>80% high</td>
</tr>
</tbody>
</table>

We randomize the order of the queries and the concepts in the individual queries. For example, the number of departments in the query is randomized.

2.4 Evaluation Metrics
We estimate the response time of single queries and elapsed time of each query sequence to see difference in two approaches, single query and query sequence. It is possible to estimate the two metrics in the following manner:

**Response Time of Each Individual Query:** We create and open a repository. All data is loaded at once before any query requests. After data loading, each query is issued ten times and we traverse the result set sequentially. After estimating average response time for individual queries, we close the repository. We repeat the above steps for all selected systems.

**Elapsed Time of Each Query sequence:** Procedure to measure elapsed time of each query sequence is almost the same as the previous one, except when it doesn’t repeat each individual query.

3. EXPERIMENT
The experiment is conducted on a 32-bit Ubuntu 7.10 machine (Ahtlon64 X2 3600+). Tester program runs in Sun JDK 1.6.02, with heap size of 1024GB and max perm size of 512MB. We use MySQL 5.0.45 for sesame-rdbms-rdfs, running at localhost.

3.1 Response Time of Single Queries
Most ontology reasoning systems show similar performance for the test queries. However, sesame-rdbms-rdfs and OWLIM show poor performance on specific queries: query 8 with sesame-rdbms-rdfs and query 9 with OWLIM. But with larger ontology size, the systems perform worse on other queries. This means that the scalability varies with respect to queries.

According to Figure 2, the tested systems differ in scalability. Therefore the performance on the ontology of one specific size cannot be taken as the performance of the system in general cases. So, when the developers select the proper system for their projects, they should perform the test with data which has similar size to their actual use cases.

Figure 3 shows the average response time grouped by dimensions. The high complexity queries take more time than the low complexity queries. On the other hand, we cannot find significant differences when selectivity of queries varies: the selectivity of queries does not affect the response time.
3.2 Elapsed Time of Query Sequences

Figure 4 shows the total execution time for each query sequence on selected systems. Pellet halt with NullPointerExceptio, not with OutOfMemoryError. In case of OWLIM, we aborted the test for QS3 and QS4 after the system ran for three hours without any output: QS3 and QS4 contain twice the number of query 8 (whose response time is extremely high in OWLIM) than QS1 and QS2. The poor performance on QS3 and QS4 had been expected.

All of the systems tested process QS1 and QS2 faster than QS3 and QS4. This is a natural result since the high complexity queries are processed more slowly as mentioned before. On the other hand, selectivity metric does not have meaningful effects on performance.

After collecting response time of all individual queries, we can calculate expected time for each query sequence. We already know the average response time of each individual query and which queries are used in each query sequence, so we calculate the expected time by multiplying the number of each query and the average execution time of the query.

Figure 5 shows the expected time and the actual elapsed time, for QS1 and QS3. It shows that some of the systems are faster in the actual cases. This means that the performance with the query sequences cannot be directly inferred from the result of individual query test.

We calculate the ratio of the total response time of query sequence to the expected time. Ratios of QS1 and QS2 on the target systems are between 0.5 and 1.1 but the processing QS3 and QS4 take far more than expected time: the ratios on the target systems are between 2.9 and 12. QS3 and QS4 have more high complexity queries, so it means that the target systems cannot run efficiently when the high complexity queries are requested randomly and frequently. Inference levels of systems do not have an effect on this tendency. Some system show less ratio changes when query sequence is changed: sesame2-mem-rdf has ratios less than 4 and it means that the system shows more stable performance than others.

4. DISCUSSIONS AND FUTURE WORKS

It is important that a benchmark result be accepted with caution. The runtime of individual tests is important but it may by the case that variation is more critical. For example, though the KAON2 seems slower than other systems in terms of runtime, coefficient of variation of response time on KAON2 is 0.4 while those of other systems are above 1.0. This means KAON2 shows relatively consistent performance over various types of queries. This uniformity is a good property when stable response time is required. The difference between expected runtime and actual runtime of query sequences shown in Figure 5 is another such example.

We have not considered data insertions or deletions in between queries; data is loaded before any query requests in the current experiment. Since dynamic insertions and deletions can affect caching and indexing, we plan to extend our experiment with query sequences that include insertions and deletions.
In our experiment, the portion of the query in each query sequence is considered but the order of the queries is not. A comparison of sequences that have same queries with different orders would be the next step in understanding the effect of the query sequence on the ontology reasoning systems.

5. ACKNOWLEDGMENTS
This study was supported by the Ministry of Information & Communications, Korea, under the Information Technology Research Center (ITRC) Support Program.

6. REFERENCES