Backward Chaining Ontology Reasoning Systems with Custom Rules

Hui Shi
College of Business
University of Southern Indiana
Evansville, IN 47712
(001) 812-4657120
hshi@usi.edu

Kurt Maly
Department of Computer Science
Old Dominion University
Norfolk, VA 23529
(001) 757-6837722
Kmaly@odu.edu

Dazhi Chong
Department of Information Technology
Old Dominion University
Norfolk, VA 23529
(001) 757-3181298
dchong@odu.edu

Gongjun Yan
College of Business
University of Southern Indiana
Evansville, IN 47712
(001) 812-2285073
gyan@usi.edu

Wu He
Department of Information Technology
Old Dominion University
Norfolk, VA 23529
(001) 757-6835008
whe@odu.edu

ABSTRACT
In the semantic web, content is tagged with "meaning" or "semantics" to facilitate machine processing and web searching. In general, question answering systems that are built on top of reasoning and inference face a number of difficult issues. In this paper, we analyze scalability issues faced by a question answering system used by a knowledge base with science information that has been harvested from the web. Using this system, we will be able to answer questions that contain qualitative descriptors such as "groundbreaking", "top researcher", and "tenurable at university x". This question answering system has been built using ontologies, reasoning systems and custom based rules for the reasoning system. Furthermore, we evaluated the performance of our optimized backward chaining engine on supporting custom rules and designed the experimental environment including scalable datasets, rule sets, query sets and metrics and compared the experimental results with other in-memory ontology reasoning systems. The results show that our developed backward chaining ontology reasoning system has better scalability than in-memory reasoning systems.

Keywords
Semantic Web; Ontology; Benchmark; Ontology Reasoning System; Backward Chaining Reasoner; Custom Rules

1. INTRODUCTION
There is a variety of structured and semi-structured information increasingly available on the Internet, which can be mined, organized, and queried in a collaborative environment. As a result, there have been more and more research on the intersections of Semantic Web, collaborative work and social media research [1-4]. For example, IkeWiki, as a semantic wiki, is mainly developed for collaborative knowledge engineering with support for different formalization [2]. A second example is SIOC (Semantically-Interlinked Online Communities), which interconnect present online communities for data integration and cross-site structural queries [1].

Despite the growing number of work in this area, there is not much research related to semantic web systems that can provide answers to qualitative queries submitted by users. In an effort to contribute to this research area, we have recently developed a semantic web system where the underlying knowledge base covers linked data about 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 is expected to answer qualitative queries such as: "Who are the groundbreaking researchers in data mining?" or "Is my record tenurable at my university?". As supporting a collaborative environment where users are allowed to customize qualitative queries requires a fast response from the reasoning system, performance and scalability have become challenging tasks for an ontology reasoning system that has to deal with custom rules required for qualitative queries.

Many knowledge bases organize information using ontologies [5-7]. An ontology is "a formal, explicit specification of a shared conceptualization" [8]. Ontologies can be used together with a reasoning engine to infer new relations. By ontology reasoning [9], we can derive implicit facts that are in ontology or in knowledge base given the explicit facts and OWL rules. By reasoning with custom rules, we can derive implicit facts that are in ontology or in knowledge base given explicitly stated facts and custom rules (user-defined rules). In paper [10], we have evaluated the performance of Jena [11], Pellet [12-13], KAON2 [14-15], Oracle 11g [16] and OWLIM [17] using representative custom rules, including transitive and recursive rules, on top of our ontology and LUBM (Leigh University Benchmark) [18]. In this paper, we have introduced our optimized backward reasoning reasoner [19] to support the science research domain where knowledge frequent changes. In addition, in this paper, we evaluate the performance of our optimized backward reasoning reasoner on supporting custom rules. And our backward chaining

WWW 2016 Companion, April 11-15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.

http://dx.doi.org/10.1145/2872518.2890521

engine outperforms the other ontology reasoning systems in scenarios where the knowledge base is subject to frequent change.

The remainder of this paper is organized as follows: In Section 2 we discuss related work on backward chaining, existing benchmarks and give a general description of OWL reasoners. In Section3, we briefly discuss the optimizations of our backward chaining reasoner. In section 4, we discuss an experimental design to evaluate our optimized backward chaining reasoner on supporting the inference of custom rules. In Section 5, we present the results, and compare the experimental results. Conclusions are given In Section 6.

2. RELATED WORK

2.1 Backward Chaining

Backward chaining is a form of goal-driven reasoning, which starts with goals from the consequents, matching the goals to the antecedents to find data satisfying the consequents. Backward-chaining does not require expensive pre-computation, but increases the cost of reasoning when answering each query [20].

4store [21] applies the RDFS rules using backward-chaining technique. Virtuoso [22] implements a mixture of forward-chaining and backward-chaining. Jena [11] supports three ways of inferencing: forward-chaining, limited backward-chaining and a hybrid of these two methods.

2.2 Ontology Reasoning Systems Supporting Custom Rules

Jena [11] supports custom rules in its proprietary format "Jena Rules". Pellet [12-13] and KAON2 [14-15] both support SWRL [23]. Oracle 11g [16] provides native inference in the database for user-defined rules. OWLIM [17] 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 aforementioned reasoning systems with regard to the complexity of the ontology and the number of triples in the ontology. 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.

3. OPTIMIZATION DETAILS & DISCUSSION

Four types of optimizations have been introduced in our algorithm for backward chaining [19]. These optimizations are:

1) the implementation of the selection function, which selects another goal left after finishing proving one goal in a rule body. Our selection function selects next goal to be resolved depends on number of free variables;

2) the upgraded substitute function, which implements the substitution of the free variables in the body clauses in one rule based on calculating a threshold that switches resolution methods;

3) the application of OLDT [24], which applies OLDT in the backward chaining process, eliminating the issue of non-termination; and

4) solving of "owl:sameAs" problem, which is a novel adaptation of "owl:sameAs" optimization in forward chaining reasoning system. We also select a representative node to represent an equivalence class of owl:sameAs URIs to avoid in a multiplication of the number of instances of variables during backward-chaining. Of these, optimizations 3 and 4 have appeared in [25, 24] whereas techniques 1 and 2 are new.

We believe that a combination of these optimization methods with our own optimizations are novel. We will describe the implementation details of ordered selection function as follows. You can find more implementation details in our previously published paper [19, 26] as well.

The proving of a rule head requires the proving of the body of a rule. The body of a rule consists of a conjunction of multiple clauses (goals). The proving of a rule head requires the proving of all the clauses. After finishing proving one goal, another goal left would be selected. This selecting process is called selection function [27]. Logically, any clause left could be selected, but, this selecting process affects the reasoning efficiency in terms of both time and scalability. Our selection function selects next goal to be resolved depends on number of free variables.

Mostly, goals with fewer free variables cost less time to be resolved than goals with more free variables, since fewer free variables means more bindings and body clauses with fewer free variables will match fewer triples.

We calculated the probability that a goal with two free variable cost more time to be resolved than a goal with one free variable. Recording the time spent on proving goals with different numbers of free variables during answering a query will help us to estimate the probability. We did not include goals with three free variables in the estimation because there are not any goals with three free variables during the reasoning process.

Let cost1 denote the time spent on resolving a goal with one free variable. Let cost2 denote the time spent on resolving a goal with two free variables.

Let n2 denote the number of goals with two free variables, n1 denote the number of goals with one free variable. Let l_i denote the number of goals with one free variable cost more time than goal i with two free variables (n2>=i>=0). Following is the formula used for calculating the probability that a goal with two free variable cost more time to be resolved than a goal with one free variable.

$$P(cost2>cost1) = Sum(l_i/n1)*(1/n2)$$

We recorded the time spent on proving goals with one free variable and two free variables during answering 14 queries from LUBM (100). Table 1 lists the probability that a goal with two free variable cost more time to be resolved than a goal with one free variable for 9 queries out of 14 queries, because there are not any goals with two free variables involved in answering those 5 queries.

Table 1. The probability that a goal with two free variable cost more time to be resolved than a goal with one free variable for 9 queries out of 14 queries

	P(cost2>cost1)
Query2	0.119
Query 4	0.079
Query 6	0.088
Query 7	0.076
Query8	0.060
Query9	0.092
Query10	0.079
Query11	0.012
Query12	0.282

As shown in Table 1, the probability that a goal with two free variable cost more time to be resolved than a goal with one free variable for queries in LUBM(100) ranges from 0.012 to 0.282. In most queries, the probability is less than 0.100, verifying that goals with fewer free variables cost less time to be resolved than goals with more free variables in most cases. Therefore, the number of free variables

4. EXPERIMENTAL DESIGN

This study is concerned about scalability of the question answering system. Research questions include: how effective is our optimized backward chaining reasoner 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 these questions, in this section we shall discuss the:

- Data (ontology classes/relations and instance data) that will be involved in querying the ontology reasoning systems, such as ontology of LUBM and corresponding datasets
- The custom rules that are going to be used. User are supposed to add the custom rules to get answers to their custom queries, such as queries to retrieve Co-authors.
- Environment and metrics to evaluate a system, such as query process time
- Evaluation procedure, such as the evaluation of our optimized backward chaining reasoner in answering queries when customized rules are added to native logic in terms of scalability and query process time.

4.1 Ontology data for the experiments

We used the LUBM [18] in our experiments. The LUBM is about concepts and relationships in a research community. For instance, concepts such as Faculty, Publication, and Organization are included in LUBM, as are properties such as advisor, publicationAuthor, and worksFor.

The size range of the datasets in our experiments is listed in Table 2. We generate 11 datasets for LUBM, which ranges from thousand to millions for our experiment.

Table 2 Size range of datasets (in triples)

Dataset1	**Dataset2**	**Dataset3**	**Dataset4**
8814	15438	34845	100838
Dataset5	**Dataset6**	**Dataset7**	**Dataset8**
624827	1272870	2522900	4109311
Dataset9	**Dataset10**	**Dataset11**	
6890949	13880279	27643953	

4.2 Custom Rule Sets and Queries

We now present the five rule sets and three corresponding query sets that we shall use in the experiments. Rule sets were defined to test basic reasoning to allow transitivity and recursion. Rule set 1 is defined for the co-authorship relation; rule set two is defined for collaborator relation; rule set three is used in queries for the genealogy of PhD advisors (transitive) and rule set 4 is used to enable queries for "good" advisors. Rule set 5 is a combination of the first 4 sets. The five rule sets are expressed as follows in Table 3. We have composed 3 query sets to use in these tests, expressed in SPARQL notation in Table 4**.**

Table 3 Rule sets expressions

Rule Set	**Definition**
1: Co-author	authorOf(?x, ?p) ∧ authorOf(?y, ?p) ⇒coAuthor(?x, ?y)
2: Collaborator	advisorOf(?x, ?y)⇒collaboratorOf (?x, ?y)
3: Research ancestor (transitive)	advisorOf(?x, ?y) ⟹ researchAncestor(?x, ?y) researchAncestor(?x, ?y) ∧ researchAncestor(?y, ?z) ⟹ researchAncestor(?x, ?z)
4: Distinguished advisor (recursive)	advisorOf(?x,?y) ∧ advisorOf(?x,?z) ∧ notEqual(?y,?z) ∧ worksFor(?x,?u) ⟹ distinguishAdvisor(?x, ?u) advisorOf(?x,?y) ∧ distinguishAdvisor(?y,?u) ∧ worksFor(?x,?d) ⇒ distinguishAdvisor(?x, ?d)
5: Combination	A combination of above 4 rule sets.

Table 4 Query sets expressed in SPARQL

Query set	**Query set**	**Expression**
Query set1	**Query 1: Co-author**	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {?x uni:coAuthor ?y.};
	Query 2: Collaborator	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {?x uni:collaboratorOf?y. };
	Query 3: Research ancestor	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {?x uni:researchAncestor ?y.};
	Query 4: Distinguished advisor	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {?x uni:distinguishAdvisor ?y. };
Query set2	**Query 1: Co-author**	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {?x uni:coAuthor ?y. ?x uni:hasName \"FullProfessor0\" }
	Query 2: Collaborator	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {?x uni: collaboratorOf ?y. ?x uni:hasName \"FullProfessor0\" };
	Query 3: Research ancestor	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {?x uni:researchAncestor ?y. ?x uni:hasName \"FullProfessor0\" };
	Query 4: Distinguished advisor	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {?x uni:distinguishAdvisor ?y. ?y uni:hasTitle \"department0\" };

Query set3	**Query 1: Co-author**	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {<http://www.d0.u0.edu/~FullProfessor0> uni:coAuthor ?y. }
	Query 2: Collaborator	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {<http://www.d0.u0.edu/~FullProfessor0> uni: collaboratorOf ?y. };
	Query 3: Research ancestor	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {<http://www.d0.u0.edu/~FullProfessor0> uni:researchAncestor ?y. };
	Query 4: Distinguished advisor	PREFIX uni:<http://swat.cse.lehigh.edu/onto/univ-bench.owl#> SELECT ?x ?y WHERE {?x uni:distinguishAdvisor <http://www.d0.u0.edu>.};

There are minor differences among the above three query sets in Table 4. Query set 1 is intended to retrieve all the pairs of relationships, for example, all the co-authors in the knowledgebase. Query set 2 is intended to retrieve partial pairs of relationships, for example, all the co-authors of researchers whose name is "FullProfessor0". Query set 3 is intended to retrieve pairs of relationships for a specific researcher/department, for example, all the co-authors of researcher <http://www.d0.u0.edu/~FullProfessor0>. Query set 2 is the query form we used in paper (Motik and Studer, 2005). For query set 2, our optimized backward chaining reasoner answers two separate queries and then applies a join operation resulting in a cross product. For query set 1 and 3, our optimized backward chaining reasoner only need to answer one single query respectively. Queries are used with the rules sets that define the properties employed in the queries. Each rule set is tested with corresponding queries in different query sets. Rule set 5 is tested with all queries.

4.3 Experimental Environment and Metrics

We have chosen Jena TDB [28] as our external storage support for our optimized backward chaining reasoner. The latest version of the systems are: Jena (2.11.0, 2013-09-18 release), Jena TDB(1.0.0, 2013-09-18 release).As backward chaining system does not require expensive up front closure computation every time the knowledge base changes, we have taken scalability and query processing time from [10] as the main metrics.

- 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.

All the experiments in this section were performed on a PC with a 2.80 GHz Intel Core i7 processor and 8 G memory, running Windows 7 Enterprise. Sun Java 1.6.0 was used for Java-based tools. The maximum heap size was set to 512M. We checked all of our results to ensure that they were complete and sound. All the timing results we present in this paper are CPU times as the knowledge base is entirely in memory.

4.4 Evaluation Procedure

Our goal is to evaluate the performance of our optimized backward chaining reasoner in terms of reasoning and querying time using custom rules. We are interested in two aspects of performance. The first aspect is scalability, which means the size of data and the complexity of reasoning. We are interested in the performance of our system as the size of the knowledge base changes from small toy size to realistic sizes of millions. The second aspect is query processing time. We are interested in the query processing time as the size of the knowledge base changes from small toy size to realistic sizes of millions. Finally we are interested in the performance of our system on a widely used model (LUBM).

5. RESULTS AND DISCUSSION

5.1 Evaluation on top of LUBM

With rule sets and query sets described in section 4.2, we have evaluated our backward chaining reasoner on 11 datasets generated from LUBM. In this section, all the datasets are stored in external storage using Jena TDB as our support. The evaluation results of our backward chaining reasoner on top of LUBM is shown in Table 5.

Table 5 Query processing time for query set1 and query set 2 on LUBM (Unit:ms)

	Query set1	Query set2	Query set1	Query set2	Query set1	Query set2	Query set1	Query set2
	Query1		Query2		Query3		Query4	
Dataset1	171	249	124	171	109	156	156	249
Dataset2	234	265	159	187	140	187	249	327
Dataset3	421	468	171	249	171	265	374	452
Dataset4	733	889	265	452	421	452	592	764
Dataset5	1669	1778	811	951	858	1029	1591	1606
Dataset6	2823	2979	1232	1435	1388	1497	2293	2386
Dataset7	4570	4836	1934	2168	2308	2324	3447	3478
Dataset8	6115	6832	2808	2839	2917	3244	4602	4655
Dataset9	10233	11044	4258	4586	4664	5116	6848	6879
Dataset10	19012	19671	7534	7960	8938	9250	11793	12074
Dataset11	37034	37752	15241	16536	18064	18392	21902	22932

We focus on two aspects of evaluation on supporting reasoning of customized rules. First is scalability. With support of external storage, our optimized backward chaining reasoner can handle up to about 30 million size dataset in our experimental environment. Second is performance in terms of query processing time. As the size of dataset increases, the query processing time scales from less than 1 second to about half minute.

As shown in Table 5, for all the datasets, query processing time of all queries from query set 1 are slightly better than query set 2 because of the minor difference that we have discussed in section 4.2. Our optimized backward chaining reasoner applies join operation to answer queries from query set 2. We believe that the other ontology reasoning systems may employ different methods in query processing to answer multi-queries.

For all the datasets, query processing time of all queries from query set 3 is less than 1 seconds. We do not present the evaluation results of query set 3 in this paper. Compared with results in paper [10] for query set 2, our optimized backward chaining reasoning system has better scalability than in-memory reasoning systems such as Kaon2 and Pellet.

5.2 Comparison between in-memory store and external storage on LUBM

We employed Jena TDB for the support of external storage in order to increase the scalability of our optimized backward chaining reasoner. We anticipate that some applications might want to balance performance and scalability. For example, for some mobile applications, performance is more important than scalability.

In this section, we compare the performance of our optimized backward chaining reasoner with and without support of external storage on all the queries from query set1 and query set2. The evaluation results are presented in Table 6.

When our optimized backward chaining reasoner runs in memory without any support of external storage, it can only handle up to 7 data sets from LUBM. Thus we only compared performance on 7 data sets. We are aware that working in memory has limitations with respect to the size of the knowledge base and the retrieved data. As Table 6 shows, our reasoner running with support of Jena TDB has twice processing time as much as running entirely in memory, that is, running entirely in memory performs better than running with support of Jena TDB. For light applications like mobile applications, in-memory version would be a better choice.

Table 6. Comparison of query processing time between in-memory store and external storage on LUBM (Unit:ms)

	TDB		In memory		TDB		In memory		TDB		In memory		TDB		In memory	
	Query set1		Query set2		Query set1		Query set2		Query set1		Query set2		Query set1		Query set2	
Dataset	Query1				Query2				Query3				Query4			
DS1	171	249	93	109	124	171	46	78	109	156	56	78	156	249	93	156
DS2	234	265	109	140	159	187	46	93	140	187	62	109	249	327	140	202
DS3	421	468	124	171	171	249	62	98	171	265	78	113	374	452	202	296
DS4	733	889	358	363	265	452	93	140	421	452	124	171	592	764	327	343
DS5	1669	1778	748	858	811	951	249	296	858	1029	296	358	1591	1606	1107	1154
DS6	2823	2979	1107	1232	1232	1435	296	374	1388	1497	390	483	2293	2386	1263	1279
DS7	4570	4836	2059	2199	1934	2168	421	530	2308	2324	639	733	3447	3478	1887	1905

6. 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" and "Who are the leading researchers in data visualization?" In this paper, we have explored the feasibility of developing a question answering system to address these queries. We have briefly introduced the optimizations of our optimized backward chaining reasoner. We have designed the experimental environment including ontologies and scalable datasets, rule sets, query sets and metrics. We have discussed the experimental results and compared the performance with other ontology reasoning systems. The evaluation results show that our optimized backward chaining reasoner has better scalability than in-memory reasoning systems such as Jena, Kaon2 and Pellet.

Generally, there are three limitations in our study. 1) The optimized backward chaining reasoner was designed for answering customized queries facing large evolving knowledge base. Although we have optimized backward chaining in terms of scalability and efficiency, the largest knowledge base we can handle is about 30 million. 2) Secondly, all our experiments were only performed "in memory" or using Jena TDB as external storage. The employment of other external storage, such as Oracle, Jena SDB, may affect the scalability and performance of the reasoner. 3) Only one types of ontology data including datasets generated from LUBM was included in the experiments, which limited the study to academic knowledge bases with specific custom rules.

We are planning to promote the optimized backward chaining reasoner by balancing the query processing time and scalability and introducing more optimization techniques. We are planning to employ other external storage (Oracle, Jena SDB, etc) and address the issues brought by the employment of other external storage. For example, the algorithm has to take now into account that it will take longer to access a triple (or a set of triples) due to having to perform I/O in a different storage. We are also planning to extend our study by using more knowledge bases from domains

other than academic during the evaluation. A wide range of knowledge bases would increase the range of custom rules and queries as well. The wide range of knowledge bases, custom rules and queries can be used in a more comprehensive evaluation, which will help improve the optimized backward chaining reasoning in terms of answering customized queries.

7. REFERENCES

[1] Breslin, J.G., Harth, A., Bojars, U. and Decker, S. (2005). Towards semantically-interlinked online communities. The Semantic Web: Research and Applications. Springer. pp. 500-514.

[2] Schaffert, S. (2006). IkeWiki: A semantic wiki for collaborative knowledge management. Enabling Technologies: Infrastructure for Collaborative Enterprises, (2006). WETICE'06. 15th IEEE International Workshops on, IEEE.

[3] Sure, Y. et al. (2002). OntoEdit: Collaborative ontology development for the semantic web: Springer.

[4] Wennerberg, P.O. (2005). Ontology based knowledge discovery in Social Networks. Final Report, JRC Joint Research Center 1-34.

[5] Bizer, C. et al. (2009). DBpedia-A crystallization point for the Web of Data. Web Semantics: Science, Services and Agents on the World Wide Web 7(3) 154-165.

[6] Doms, A. and Schroeder, M. (2005). GoPubMed: exploring PubMed with the gene ontology. Nucleic acids research 33(suppl 2) W783-W786.

[7] Mars, N.J. (1995). Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing 1995: Ios Press.

[8] Gruber, T.R. (1993). A translation approach to portable ontology specifications. Knowledge acquisition 5(2) 199-220.

[9] Horrocks, I. and Sattler, U. (2001). Ontology reasoning in the SHOQ (D) description logic. IJCAI.

[10] Shi, H., Maly, K., Zeil, S. and Zubair, M. (2011). Comparison of Ontology Reasoning Systems Using Custom Rules. International Conference on Web Intelligence, Mining and Semantics. Sogndal, Norway.

[11] The Apache Software Foundation. (2014)a. Apache Jena [online]. Available at: http://jena.apache.org/ [Accessed February,16 2016].

[12] Clark & Parsia. (2010). Pellet:The Open Source OWL2 Reasoner [online]. Available at: http://clarkparsia.com/pellet/ [Accessed October,13 2010].

[13] Sirin, E. et al. (2007). Pellet: A practical owl-dl reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2) 51-53.

[14] Information Process Engineering (IPE), Institute of Applied Informatics and Formal Description Methods (AIFB) and (IMG), I.M.G. (2014). KAON2-Ontology Management for the Semantic Web [online]. Available at: http://kaon2.semanticweb.org/ [Accessed February 16 2016].

[15] Motik, B. and Studer, R. (2005). KAON2–A Scalable Reasoning Tool for the Semantic Web. Proc. 2nd ESWC. Heraklion, Greece.

[16] Xavier Lopez and Das, S. (2010). Semantic Technologies in Oracle Database 11g Release 2: Capabilities, Interfaces, Performance.

[17] Kiryakov, A., Ognyanov, D. and Manov, D. (2005). OWLIM–a pragmatic semantic repository for OWL. Web Information Systems Engineering 3807/2005 182-192.

[18] Guo, Y., Pan, Z. and Heflin, J. (2005). LUBM: A benchmark for OWL knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web 3(2-3) 158-182.

[19] Shi, H., Maly, K., & Zeil, S. (2014). A scalable backward chaining-based reasoner for a semantic web. International Journal on Advances in Intelligent Systems, 7(1-2), 23-38.

[20] Urbani, J., Piro, R., van Harmelen, F. and Bal, H. (2013). Hybrid reasoning on OWL RL. Semantic Web.

[21] Harris, S., Lamb, N. and Shadbolt, N. (2009). 4store: The Design and Implementation of a Clustered RDF store. Proceedings of the 5th International Workshop on Scalable Semantic Web Knowledge Base Systems. 81-96.

[22] Erling, O. and Mikhailov, I. (2009). RDF Support in the Virtuoso DBMS. Networked Knowledge-Networked Media 7-24.

[23] Horrocks, I. et al. (2004). SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission 21.

[24] Tamaki, H. and Sato, T. (1986). OLD resolution with tabulation. Third international conference on logic programming. Springer. 84-98.

[25] Ontotext. (2013). owl-sameAs-optimization [online]. Available at: http://graphdb.ontotext.com/documentation/standard/sameas-optimisation.html [Accessed February 16 2016].

[26] Shi, H., Maly, K., & Zeil, S. (2014, June). Optimized backward chaining reasoning system for a semantic web. In Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14) (p. 34). ACM.

[27] Kowalski, R. and Kuehner, D. (1972). Linear resolution with selection function. Artificial Intelligence 2(3) 227-260.

[28] The Apache Software Foundation. (2014)b. Apache Jena-TDB [online]. Available at: http://jena.apache.org/documentation/tdb/.[Accessed February 16 2016].