



LarKC

*The Large Knowledge Collider:
a platform for large scale integrated reasoning and Web-search*

FP7 – 215535

D5.5.1 Measurable Targets for Scalable Reasoning

(formerly: Definition of validation goals for the prototyping phase)

Coordinator: Atanas Kiryakov, Onto

Document Identifier:	LarKC/2008/D5.5.1 /v1.0
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	1.0
Date:	30.06.2008
State:	Final
Distribution:	Public



EXECUTIVE SUMMARY

Defining adequate performance targets is crucial for the development of scalable semantic repositories that combine VLDB management with reasoning. I start with an analysis of the relevant tasks (especially, loading and query answering), performance criteria and problem dimensions (e.g. speed, complexity, size). The major contributions are the analysis of the state of the art in scalable reasoning and the definition of measurable performance targets to be met by year 2010. The analysis is based on published results of several of the most scalable engines: ORACLE, AllegroGraph, DAML DB, Openlink Virtuoso, and BigOWLIM. The targets are defined with respect to two of the currently most popular performance measuring sticks: the LUBM repository benchmark (and its UOBM modification) and the OWL version of UNIPROT – the richest database of protein-related information.

This is a second updated and extended version of the paper; the initial version which was published Nov 2007. It contains new results for most of the tools as well as some additional analysis. Most notably, while the best scalability achievements in the end of year 2007 were in the range up to 1 billion RDF statements, now we have results from several tools that approach the 10 billion statements threshold. Few of the leading providers already have distributed implementations.

NOTE: this is a pre-final version of this update of the white paper. The final one will be published in July 2008 and disseminated appropriately.



DOCUMENT INFORMATION

IST Project Number	FP7 - 215535	Acronym	LarKC
Full Title	The Large Knowledge Collider: a platform for large scale integrated reasoning and Web-search		
Project URL	http://www.larkc.eu/		
Document URL			
EU Project Officer	Stefano Bertolo		

Deliverable	Number	D5.5.1	Title	Definition of validation goals for the prototyping phase
Work Package	Number	WP5	Title	The Collider platform

Date of Delivery	Contractual	M 02	Actual	30.06.2008 (delayed)
Status	version 1.0		final x	
Nature	prototype <input type="checkbox"/> report x dissemination <input type="checkbox"/> other <input type="checkbox"/>			
Dissemination level	public x consortium <input type="checkbox"/>			

Authors (Partner)				
Responsible Author	Name	Atanas Kiryakov	E-mail	naso@sirma.bg
	Partner	Onto	Phone	












Abstract (for dissemination)	<p>Defining adequate performance targets is crucial for the development of scalable semantic repositories that combine VLDB management with reasoning. I start with an analysis of the relevant tasks (especially, loading and query answering), performance criteria and problem dimensions (e.g. speed, complexity, size). The major contributions are the analysis of the state of the art in scalable reasoning and the definition of measurable performance targets to be met by year 2010. The analysis is based on published results of several of the most scalable engines: ORACLE, AllegroGraph, DAML DB, Openlink Virtuoso, and BigOWLIM. The targets are defined with respect to two of the currently most popular performance measuring sticks: the LUBM repository benchmark (and its UOBM modification) and the OWL version of UNIPROT – the richest database of protein-related information.</p> <p>This is a second updated and extended version of the paper; the initial version which was published Nov 2007. It contains new results for most of the tools as well as some additional analysis. Most notably, while the best scalability achievements in the end of year 2007 were in the range up to 1 billion RDF statements, now we have results from several tools that approach the 10 billion statements threshold. Few of the leading providers already have distributed implementations.</p>
Keywords	performance, scalable, semantic, reasoning, measurable performance targets



Version Log			
Issue Date	Rev. No.	Author	Change
08.05.2008	0.1	Atanas Kiryakov (Onto)	Initial draft of the paper
19.06.2008	0.2	Atanas Kiryakov (Onto)	Paper ready for internal quality assessment
27.06.2008	0.3	Uwe Keller (UIBK)	Quality review feedback
30.06.2008	0.4	Atanas Kiryakov (Onto)	Quality review comments included. Paper ready for submission
01.07.2008	1.0	Alice Carpentier (UIBK), Georgina Gallizo (HLRS)	Deliverable ready for submission in LarKC template



PROJECT CONSORTIUM INFORMATION

Participant's name	Partner	Contact
Semantic Technology Institute Innsbruck, Universitaet Innsbruck	 	Prof. Dr. Dieter Fensel, Semantic Technology Institute (STI), universitaet Innsbruck, Innsbruck, Austria, E-mail: dieter.fensel@sti-innsbruck.at
AstraZeneca AB		Bosse Andersson AstraZeneca Lund, Sweden Email: bo.h.andersson@astrazeneca.com
CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA		Emanuele Della Valle, CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA, Milano, Italy, Email: emanuele.dellavalle@cefriel.it
CYCORP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O.		Michael Witbrock, CYCORP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O., Ljubljana, Slovenia, Email: witbrock@cyc.com
Höchstleistungsrechenzentrum, Universitaet Stuttgart		Georgina Gallizo, Höchstleistungsrechenzentrum, Universitaet Stuttgart, Stuttgart, Germany, Email: gallizo@hlrs.de
MAX-PLANCK GESELLSCHAFT ZUR FOERDERUNG DER WISSENSCHAFTEN E.V.		Dr. Lael Schooler Max-Planck-Institut für Bildungsforschung Berlin, Germany Email: schooler@mpib-berlin.mpg.de
Ontotext Lab, Sirma Group Corp		Atanas Kiryakov, Ontotext Lab, Sofia, Bulgaria Email: atanas.kiryakov@sirma.bg
SALTLUX INC.		Kono Kim, SALTLUX INC, Seoul, Korea, Email: kono@saltlux.com
SIEMENS AKTIENGESELLSCHAFT		Dr. Volker Tresp, SIEMENS AKTIENGESELLSCHAFT, Muenchen, Germany, E-mail: volker.tresp@siemens.com
THE UNIVERSITY OF SHEFFIELD		Prof. Dr. Hamish Cunningham, THE UNIVERSITY OF SHEFFIELD Sheffield, UK, Email: h.cunningham@dcs.shef.ac.uk






<p>VRIJE UNIVERSITEIT AMSTERDAM</p>		<p>Prof. Dr. Frank van Harmelen, VRIJE UNIVERSITEIT AMSTERDAM, Amsterdam, Netherlands, Email: Frank.van.Harmelen@cs.vu.nl</p>
<p>THE INTERNATIONAL WIC INSTITUTE, BEIJING UNIVERSITY OF TECHNOLOGY</p>		<p>Prof. Dr. Ning Zhong, THE INTERNATIONAL WIC INSTITUTE, Mabeshi, Japan, Email: zhong@maebashi-it.ac.jp</p>
<p>INTERNATIONAL AGENCY FOR RESEARCH ON CANCER</p>	 <p>International Agency for Research on Cancer Centre International de Recherche sur le Cancer</p>	<p>Dr. Paul Brennan, INTERNATIONAL AGENCY FOR RESEARCH ON CANCER, Lyon, France, Email: brennan@iarc.fr</p>



TABLE OF CONTENTS

1. PROLOGUE	8
2. INTRODUCTION	9
2.1. REASONING STRATEGIES	9
2.2. RDF, NAMED GRAPHS AND QUADS	10
2.3. OWL LAYERING AND VARIATIONS	12
3. SEMANTIC REPOSITORY TASKS AND PERFORMANCE FACTORS	15
3.1. PERFORMANCE DIMENSIONS.....	15
3.2. NOWADAYS MEASURING STICKS: LUBM, UNIPROT, DBPEDIA	16
3.3. FULL-CYCLE BENCHMARKING.....	17
4. STATE OF THE ART OF SCALABLE REASONING.....	19
4.1. SCALABILITY OF DL REASONING.....	19
<i>UOBM Benchmark Results</i>	19
<i>SHER and ABox Summarization</i>	21
4.2. SYSTEMS AND DATASETS.....	23
<i>AllegroGraph</i>	23
<i>BigOWLIM</i>	24
<i>DAML DB</i>	25
<i>ORACLE</i>	25
<i>Virtuoso</i>	26
<i>YARS2</i>	26
4.3. LOADING PERFORMANCE	27
4.4. QUERY EVALUATION PERFORMANCE	29
4.5. 64-BIT SCALABILITY NOTES.....	30
4.6. DISTRIBUTED REASONING NOTES.....	30
5. PERFORMANCE TARGETS.....	31
6. CONCLUSION	32
7. REFERENCES	33



1. Prologue

This document presents initial work on defining measurable performance targets regarding scalable reasoning. The motivation came from the need for defining success criteria for project LARKC (Large Knowledge Collider). LARKC focuses on development of reasoning methods and infrastructure that can match the performance, scalability, and behavior of the web search engines: managing billions of documents/facts; real-time handling of large masses of users; incomplete and imperfect results; relevance ranking. An important issue is that the reasoning scheme and the infrastructure within LARKC have to allow for balancing and controlling two parameters: cost (computational resources allocated) and level of satisfaction.

The second incentive was Ontotext's involvement in project RASCALLI, where we develop RDF engine with support for both logical reasoning and connectionist style spreading activation (on top of "weighted" RDF statements). We call this engine RDF Mind and use it to represent the mind of RASCALLI agents, "living" on Internet, in a way that resembles as much as practical the analogous cognitive functions of humans. In this context, there are two reasoning set-ups:

- "Working memory" - requires database-style (sound and complete) management of datasets of up to 100 million statements, and
- "Long-term memory" - requires incomplete reasoning against billions of statements, with requirements quite similar to those for web-scale reasoning in LARKC.

Last but not least, I wanted to define some criteria for the development of Ontotext's semantic repository, named OWLIM. Competition is an important stimulus for development, but it works efficiently only in mature "markets". Because of the early stage of development and the heterogeneity of the field, the leading semantic repository providers often appear without serious competition in specific "disciplines", racing against clock over considerable periods. I believe that defining adequate performance targets is crucial for the development of scalable semantic repositories.

My goal is to provide a simplistic view on some aspects of the performance of semantic repositories, which allows for objective measurement. The reader should consider that the criteria defined here are shallow or ignorant with regards to substantial aspects of the reasoning tasks in focus. Most notably, no or insufficient attention is paid to:

- Accuracy of the relevance ranking;
- Reasoning with non-tractable logical fragments.

The state of the art analysis and benchmarking work included in the original version was done in projects RASCALLI, TAO, and TripCom. The updated version is produced in the course of the work on project LarKC (<http://www.larkc.eu/>) and represents deliverable D5.5.1 "Definition of validation goals for the prototyping phase".



2. Introduction

Let us call “semantic repository” (SR) a tool, which combines the functionality of an RDF-based DBMS and an inference engine. A semantic repository can store data and evaluate queries, regarding the semantics of ontologies and metadata schemata.

Performance and scalability targets for SR are specified here based on analysis of the state-of-the-art engines. One shall consider however that direct comparison between a “classical” SR and a web-scale-and-spirit one is not feasible, because:

- Comparison between complete and incomplete reasoning and query evaluation has limited utility. Given a “tough” query to evaluate, an incomplete SR can consume as much time as it is allowed – more time delivers more and better quality results; in many cases there would be no upper limit of the time that can be used;
- The classical inference engines do not support relevance ranking of the results. Relevance ranking can be computationally expensive task, especially when context relevance and personalization are taken into account.

2.1. Reasoning Strategies

Many of the scalable semantic repositories nowadays perform reasoning that is based (more or less directly) on logical entailment rules. The two principle strategies for rule-based inference are, as follows:

- **Forward-chaining:** to start from the known facts (the explicit statements) and to perform inference in an inductive fashion. The goals of such reasoning can vary: to compute the *inferred closure*¹; to answer a particular query; to infer a particular sort of knowledge (e.g. the class taxonomy).
- **Backward-chaining:** to start from a particular fact or a query and to verify it or get all possible results (bindings for the free variables), using deductive reasoning. In a nutshell, the reasoner decomposes (or transforms) the query (or the fact) into simpler (or alternative) facts, which are available in the knowledge base (KB) or can be proven through further recursive transformations.

Both of these strategies have well-known strong and weak points. Hybrid strategies are also possible and have proven to be efficient in many contexts.

Let us imagine a repository which performs total forward-chaining, i.e. after update to the KB the inferred closure is updated and kept available for query evaluation and retrieval. This strategy is known as *materialization*. In order to avoid ambiguity with various partial materialization approaches, let us call such inference strategy *total materialization*.

The principle advantages and disadvantages of the total materialization are discussed at length in [2]; here we provide just a short summary of them:

- Upload/storage/addition of new facts is relatively slow, because the repository is extending the inferred closure after each transaction for modification. In fact, all the reasoning is performed during the upload;
- Deletion of facts is also slow, because the repository should remove from the inferred closure all the facts which are not true any longer;
- The maintenance of the inferred closure usually requires considerable additional space (RAM, disk, or both, depending on the implementation);

¹ *Inferred closure* is defined here as follows: the extension of a KB (or a graph of RDF triples) with all the implicit facts (triples), which could be inferred from it, based on the enforced semantics.



- Query and retrieval are fast, because no deduction, satisfiability checking, or other sorts of reasoning are required. Query evaluation becomes computationally comparable to the same task for relation database management systems (RDBMS).

Apart from the principle advantages and disadvantages of the reasoning strategies, their applicability is often pre-determined by the complexity of the semantics (ontology language) that has to be supported. Imagine a dataset, where large groups of individuals are interconnected though a transitive and symmetric property, e.g. fellow-citizen (as in [25]). If each citizen of a city with 100 thousand citizens is connected explicitly, say, with two others, there will be 200 thousand explicit fellow-citizen statements in the dataset. However, the inferred closure will contain about 10 billion statements! Obviously, total materialization can face troubles even with tractable logical formalisms.

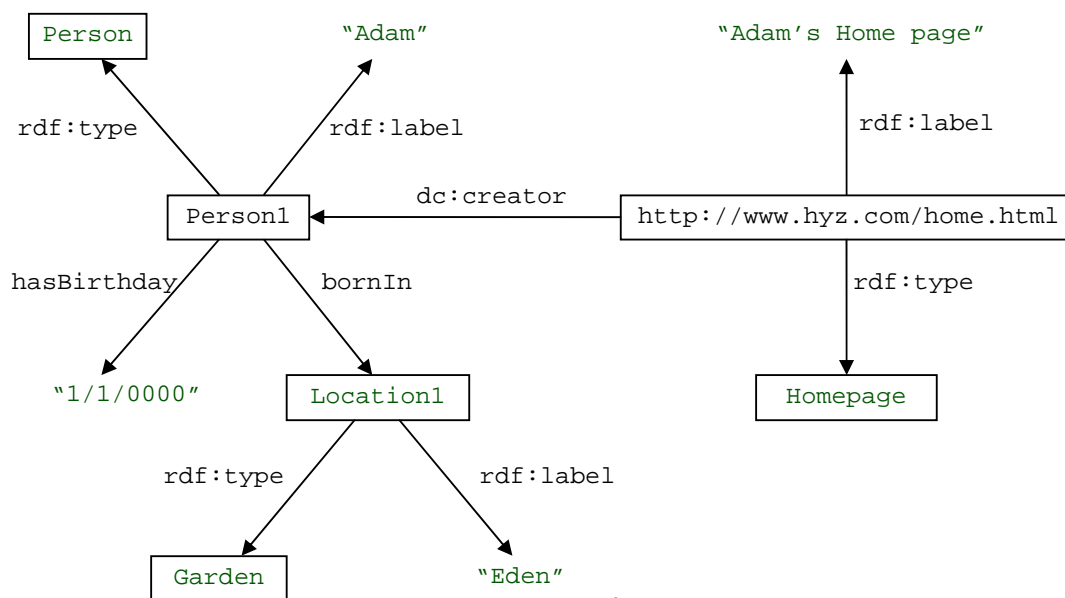
2.2. RDF, Named Graphs and Quads

A family of mark-up and KR standards were developed under W3C-driven community processes, as a basis for the Semantic Web. RDF, [24], is a metadata representation language, which serves as a basic data-model for the Semantic Web. It allows resources to be described through relationships to other resources and literals. The resources are defined through URIs (unified resource identifiers, as in XML; e.g. URL). The notion of resource is virtually unrestricted; anything can be considered as a resource and described in RDF: from a web page or a picture published on a web to concrete entities in the real world (e.g. people, organisations) or abstract notions (e.g. the number Pi and the musical genre Jazz). Literals (again as in XML) are any concrete data values e.g. strings, dates, numbers, etc. The main modelling block in RDF is the statement – a triple <Subject, Predicate, Object>, where:

- **Subject** is the resource, which is being described;
- **Predicate** is a resource, which determines the type of the relationship;
- **Object** is a resource or a literal, which represents the “value” of the attribute.

A set of RDF triples can be seen as a graph, where resources and literals are nodes and each statement is represented by a labelled arc (the Predicate or relation), directed from the Subject to the Object. So-called blank nodes can also appear in the graph, representing unique anonymous resources, used as auxiliary nodes or existentially quantified variables. A sample graph, which describes a web page, created by a person called Adam, can be seen in Figure 1.

Figure 1. RDF Graph Describing Adam and His Home Page





Resources can belong to (formally, be *instances* of) classes – this can be expressed as a statement through the **rdf:type** system property as follows: **<resource, rdf:type, class>**. Two of the system classes in RDF are **rdfs:Class** and **rdf:Property**. The instances of **rdf:Class** are resources which represent classes, i.e. those resources which can have other resources as instances. The instances of **rdf:Property** are resources which can be used as predicates (relations) in triple statements.

MacGregor suggests another approach to efficiently link context to statement, by extending the RDF data model with a fourth element, [28]. The quadruple are commonly referred as quads and are represented as: **<S,P,O,C>**. The semantics of the fourth element (named “context”) is not strictly defined and has different meaning in the various quadruple stores. Different authors suggest different application of this fourth element: data source, identifier of a statement or model id. Named RDF graph data models defined by Carroll, [4], refines stricter semantic over the fourth element by defining it as *RDF graph name*, which could not be of type blank node. The named RDF graphs are designed to be disjoint from each other and define that the blank-nodes could not be shared across different graphs.

Although not explicitly stated, the philosophy of the named graphs (NG) is that, when joined, those form an RDF multi-graph, i.e. when one and the same triple appears in two graphs, it should be counted and manipulated as two separate arcs.

In SPARQL, [34], *RDF Dataset* is defined as

$$\{ \mathbf{G}, (\langle \mathbf{U}_1 \rangle, \mathbf{G}_1), (\langle \mathbf{U}_2 \rangle, \mathbf{G}_2), \dots (\langle \mathbf{U}_n \rangle, \mathbf{G}_n) \}$$

where **G** and each **G_i** are RDF graphs, and each **<U_i>** is a distinct IRI². The pairs (**<U_i>**, **G_i**) are called *named graphs*, where **<U_i>** is the name of graph **G_i**. **G** is called *default graph* – it contains all triples, which belong to the dataset, but not to any specific named graph. As we can see, SPARQL adds the notion of “default graph” which is not present in [4].

ORDI SG³ specifies a data model that is defined as an extension of the RDF, introducing Named Graphs (as discussed above) and *triple sets*. A Triple set is a new element in RDF statements, introduced in the ORDI SG model to describe the association between a statement and an identifiable group of statements. Given the above extensions, the atomic entity of the triple set model is a quintuple:

$$\langle \mathbf{S}, \mathbf{P}, \mathbf{O}, \mathbf{NG}, \{ \mathbf{TS}_1, \dots, \mathbf{TS}_n \} \rangle$$

where **NG** is the context of a statement (i.e. the named graph) and **{TS₁, ... TS_n}** is a set of identifiers of the triple sets to which the contextualized statement **<S, P, O, NG>** is associated. In other words, each statement (from each graph) can be member of multiple triple sets.

While [4] determines the semantics of the named graphs to some extent, there are some aspects which remain uncovered there. The ORDI data-model specification, [26], provides more formal definition of the semantics of the named graphs and differentiates them from triple sets as follows:

- An RDF dataset can be seen as RDF multi-graph, where, for instance, one and the same triple **<S, P, O>** can appear as two arcs if it belongs to two named graphs, e.g. if the quadruples **<S, P, O, NG₁>** and **<S, P, O, NG₂>** are part of the dataset.
- When a statement is removed from a specific named graph, the corresponding arc from the multi-graph of the dataset is removed;
- When a statement is removed from (or disassociated with) a triple set, the corresponding arc is excluded from the set of the members of the triple set, but it is not deleted from the graph.

Intuitively, one can say that named graphs “own” the triples which belong to them, while triple sets are just tagging or grouping mechanism.

² IRI is the internationalized form of URI, <http://www.ietf.org/rfc/rfc3987.txt>

³ <http://www.ontotext.com/ordi/>, an ontology management and data integration middleware framework



2.3. OWL Layering and Variations

This subsection provides an overview of some variations of metadata and knowledge representation standards, relevant to the subject of scalable reasoning in web contexts. In my opinion, one of the major challenges to building scalable Semantic Web infrastructure is the expressivity of the underlying standards: RDFS (define in [1] and [24]) and OWL (defined in [5]). The problem is that inference scalability and semantic compatibility were not primary concerns during the definition of some of the language layers (e.g. OWL Lite).

Even though RDFS can be considered a very simple Knowledge Representation (KR) language, it is already a challenging task to implement a repository for it, especially one that provides performance and scalability comparable to those of relational database management systems (RDBMS). Going up the “layers” of the Semantic Web specifications stack, the challenges for the repository engineers are getting more and more serious. Even the simplest official dialect of OWL (OWL Lite) is based on Description Logic (DL), whose complexity renders sound and complete reasoning against large knowledge bases (KB) theoretically impossible. Furthermore, the semantics of OWL Lite and OWL DL are incompatible with that of RDF(S), see [16]. This causes lack of “backward compatibility”. Imagine the situation when an application, using RDFS schemata and RDFS-compliant repository, should be “upgraded” to OWL. The evolution should start with replacement of RDFS schemata with OWL ontologies and adoption of a repository supporting (the corresponding part of) OWL. Even the most direct translation (re-labeling the `rdfs:Class` to `owl:Class`) of the schema, can lead to different inference and to inconsistencies.

Logic programming (LP) is a common name used for rule-based logical formalisms such as PROLOG, Datalog, and F-Logic. OWL DLP is a non-standard dialect, offering a promising compromise between expressive power, efficient reasoning, and RDFS compatibility. OWL DLP is defined in [16] as the intersection of the expressivity of OWL DL and LP. In fact, OWL DLP is defined as the most expressive sub-language of OWL DL, which can be mapped to Datalog. OWL DLP is simpler than OWL Lite. The alignment of its semantics to the one of RDFS is easier, in comparison with the OWL Lite and OWL DL dialects. Still, this mapping can only be achieved through the enforcement of some additional modelling constraints and transformations. A broad collection of resources related to OWL DLP can be found at <http://logic.aifb.uni-karlsruhe.de/>.

In [37], ter Horst defines RDFS extensions towards rule support and describes a fragment of OWL, more expressive than DLP. He introduces the notion of R-entailment of one (target) RDF graph from another (source) RDF graph on the basis of a set of entailment rules R. R-entailment is more general than the D-entailment used by Hayes, [20], in defining the standard RDFS semantics. Each rule has a set of premises, which conjunctively define the body of the rule. The premises are “extended” RDF statements, where variables can take any of the three positions.

Horst extends and modifies the D-entailment rules from [20] in two steps as follows: D*-entailment adds entailment support for literal data-types; pD*-entailment adds rules which provide partial support for some OWL primitives, namely: `FunctionalProperty`, `InverseFunctionalProperty`, `SymmetricProperty`, `TransitiveProperty`, `sameAs`, `inverseOf`, `equivalentClass`, `equivalentProperty`, `onProperty`, `hasValue`, `someValuesFrom`, `allValuesFrom`, `differentFrom`, `disjointWith`. The last two primitives are supported through inconsistency rules which fire in case of the so-called P-clashes. It is important to acknowledge that some of the primitives are only partially supported; the standard OWL entailments related to `someValuesFrom` and `allValuesFrom` are supported only in one of the directions (i.e. there is no full support for the iff-semantics of these OWL primitives).

I refer to this extension of RDFS as “OWL Horst”. As outlined in [37], it has several important characteristics:

- It is a proper (backward-compatible) extension of RDFS. In contrast to OWL DLP, it puts no constraints on the RDFS semantics. The widely discussed meta-classes (classes as instances of other classes) are not disallowed in OWL Horst. It also does not enforce the “unique name” assumption;



- Unlike the DL-based rule languages, like SWRL, [22] and [29], R-entailment provides a formalism for rule extensions without DL-related constraints;
- Its complexity is lower than the one of SWRL and other approaches combining DL ontologies with rules, see section 5 of [37].

My observation is that most of the languages supported by the contemporary scalable semantic repositories are very similar to OWL-Horst, even when additional or alternative rules are used. Although additional rules make entailment more expensive, “safe” extensions raise the complexity by a constant. According to [37], a sufficient condition for remaining in the same class of complexity is that the rules should not introduce new blank nodes in the knowledge base. My experience shows, that for real-world OWL ontologies and datasets, it is sufficient to enforce the above conditions only with respect to the rules, dealing with ABox reasoning (e.g. with instances). For instance, introducing a new blank node, which represents an auxiliary OWL restriction (or class), does not actually lead to exponential growth of the inferred facts.

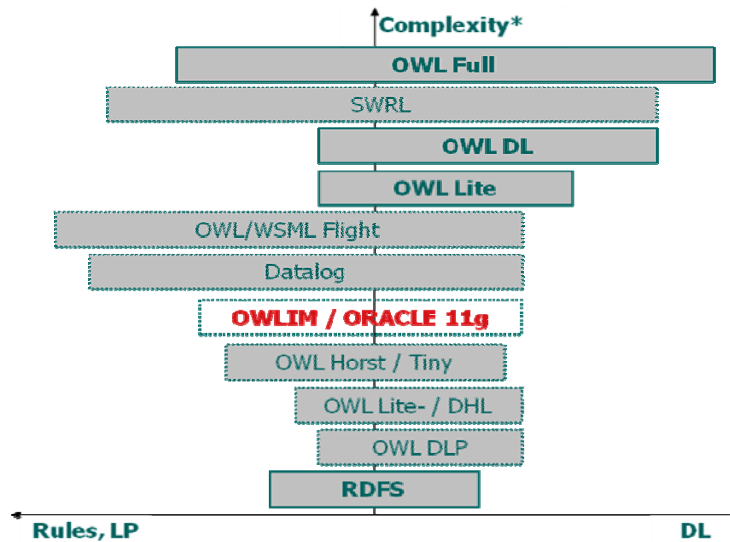


Figure 2. OWL Layering Map

Figure 2 presents a simplified map of the expressivity or complexity of a number of OWL-related languages, as well as their bias towards DL and LP semantics. The figure provides rough evidence about the expressivity of the languages, based on the complexity of entailment algorithms for them. Direct comparison between these different languages is impossible in many of the cases. For instance, Datalog is not simpler than OWL DL, it introduces a different type of complexity.

I also present on this figure the complexity supported by two of the state-of-the-art scalable repositories. Although the concrete semantics supported by those engines is not discussed here in detail, I believe it provides useful reference. The figure presents the positioning of OWLIM’s reasoning capabilities, in particular its **owl-max** rule-set, as defined in [31]. Based on the descriptions from [41], ORACLE 11g’s **OWL Prime** rule-set defines an OWL fragment similar to OWLIM’s **owl-max**. I have to note, however, that:

- The two languages are not the same; **OWL Prime** seems to be a bit simpler;
- ORACLE 11g allows usage of an external engine for TBox reasoning and materialization on the schema/ontology part of the dataset.

The results on query evaluation completeness, provided in [40], suggest that OWL Prime, combined with Pellet, [3], can answer completely all queries in the LUBM benchmark, [17]. This is an indication that OWL Prime has complexity comparable to this of OWLIM’s **owl-max** rule-set, regarding ABox reasoning. The results reported in [36] suggest that DAML DB supports a language of comparable complexity as well.



FP7 - 215535
Deliverable 5.5.1

An extended discussion on the subject can be found at:
http://www.ontotext.com/inference/rdfs_rules_owl.html.



3. Semantic Repository Tasks and Performance Factors

The performance of a “classical” semantic repository (SR) has to be measured at least for the following tasks:

- **Data loading**, including storing and indexing of both instance data and ontologies. The performance depends on several factors:
 - **Materialization** – whether and to what extent forward-chaining is performed at load time;
 - **Complexity of the data model** – some SR employ extended RDF data models, e.g. including support for named graphs. Richer data-models are more “expensive” to build and maintain.
- **Query evaluation**. There are several factors which affect the time and memory space, or more generally, the computing resources, necessary for this task:
 - **Deduction** – whether and to what extent backward-chaining is involved;
 - **Size of the result-set** – fetching large result-sets can take considerable time.
 - **Query complexity** – the number of the constraints (e.g. triple-pattern joins), the semantics of the query (e.g. negation-related clauses), the usage of operators which are tough to support through indexing (e.g. LIKE).

Searching for a meaningful measure of the evaluation speed across a set of diverse queries, I introduce a metric called query-time-per-result (QTPR). It is calculated as query evaluation time divided by the number of the results. Although QTPR is still not the perfect query performance measure, I claim that it is much more useful than the query evaluation time, taken as it is. The least to say is that it allows for accounting the fetching time, which has a serious impact for large result sets. It also allows getting a meaningful average measure across query sets where the number of results varies in orders of magnitude.

Another measure of query performance could be the response time, in the sense of time elapsed before, the first result is returned. I do not refer to this measure in the rest of this analysis, as long as such data is not generated by the popular benchmark frameworks and such data is not publicly available. Still, it is highly relevant to a wide class of applications and should be considered in the course of performance evaluation of semantic repositories.

“Modifications” to the data and/or schemata, e.g. updating and deleting values, represent another important class of the tasks performed against a SR. I am not discussing modifications here, because their complexity and importance can change considerably across applications and usage patterns.

Similarly, transaction size and level of isolation supported may also have serious impact on the performance. Although many semantic repositories provide some extent of transaction isolation, it is usually less comprehensive than this of mature RDBMS. The same holds for handling large numbers of simultaneous users – while most of the SR can handle multiple users their performance under serious load is not well studied. For this reason, transaction isolation and number of users are not discussed further in the paper.

3.1. Performance Dimensions

Reasoning performance targets are defined in terms of speed, in the sense of either throughput or response time. There are several parameters which affect the speed of a semantic repository for both loading and query evaluation:

- **Scale** – the size of the repository in terms of number of facts/triples. For engines using forward-chaining and materialization, the volume of the data they have to play with includes the inferred

facts/triples. Note that the RDF/OWL representation of Wordnet⁴ expands after materialization from 1.9 million statements to 8.2 million;

- **Schema and data reasoning complexity** – the complexity of the ontology/logical language, the specific ontology (or schema), and the dataset. E.g. a highly interconnected dataset, with long chains of transitive properties, can appear much more challenging for reasoning compared to another dataset, even when both are encoded against one and the same ontology;
- **Hardware and software setup** – the performance can vary considerably depending on the version/configuration of the compiler/virtual machine, the operating system, the configuration of the SR itself and the hardware configuration, of course.

The above analysis demonstrates that one should be realistic about the expectations regarding the utility of a simple target for the performance of a SR. Each specific benchmark provides information about the performance of an engine for a specific combination of parameters; it presents a low-dimensional section (the benchmark results) of the multi-dimensional “body” of the SR performance. A benchmark designed to measure the performance of a SR, dealing with single user’s calendar and contact data on a smart phone, is not likely to be useful for evaluating another SR, which should allow hundreds of users to investigate billions of facts in the life science domain. The same holds for the RDBMS – one can hardly define a single simple performance target to measure their performance on. It took decades for the RDBMS community to come up with standard benchmarks like TPC (<http://www.tpc.org/>). Yet, the major RDBMS vendors have the policy to avoid competitive performance benchmarks. Today, TPC is primarily used by hardware and OS vendors to compare the performance and the cost efficiency of server configurations.

3.2. *Nowadays Measuring Sticks: LUBM, UNIPROT, DBPedia*

Here I present some benchmarks and datasets which are most commonly used as measuring sticks for the performance of semantic repositories. I refer to these benchmarks in the state of the art analysis (section 4) and in the definition of the scalability targets (section 5).

The Lehigh University Benchmark (LUBM) is an outstanding benchmark for OWL repository scalability, defined in [17]. It employs synthetically generated datasets using a fixed OWL ontology of university organizations. The complexity of the language constructs used is between the one of OWL Horst and OWL DLP, as shown on Figure 1. The LUBM benchmark defines 14 queries that are used to check the query evaluation correctness and speed of repositories that have loaded a given dataset. The biggest standard dataset is LUBM(50) (i.e. it contains synthetic data for 50 universities); its size is 6.8 million explicit statements, and if written to the hard disk (in 1000 RDF/XML files), it takes 600Mbytes. For the purposes of scalability measurements many groups have used the LUBM generator to create bigger datasets, e.g. LUBM(1000) and LUBM(8000) which contain respectively 133 and 1086 million explicit statements.

UOBM, [25], is a benchmark that puts on test scalable ABox (instance data) reasoning against a relatively inexpressive DL ontology. UOBM is a further development of the LUBM benchmark; it uses the same evaluation framework (i.e. Java libraries), but provides alternative ontology, knowledgebase and queries, which allow for:

- More comprehensive coverage and usage of the OWL Lite and OWL DL semantics.
- Additional connections across the datasets for different universities – this modification assures higher level of connectivity in the RDF graph, which provides a more realistic and interesting test case.

The UOBM benchmark includes two distinct datasets for OWL Lite and OWL DL, each of which includes data collections for 1, 5 or 10 universities, named respectively: Lite-1, Lite-5, Lite-10, DL-1, DL-5, and DL-10. No dataset generator for UOBM is publicly available at present. The Lite version of the test contains 13

⁴ Wordnet is the most popular lexical knowledge base, <http://www.w3.org/TR/wordnet-rdf/>. Results on loading its RDF/OWL representation in OWLIM are presented in [31].



queries; the DL version adds two more. Both the Lite and the DL variants of UOBM require considerably more complex reasoning to be performed by a semantic repository in order to provide correct answers to the queries.

UniProt⁵ (Universal Protein Resource) is the world's most comprehensive and most popular database of information on proteins, created by joining the information contained in several other resources (Swiss-Prot, TrEMBL, and PIR). UniProt RDF, [39], is an RDF representation of the database with respect to OWL ontology, expressed in a sub-language of OWL Lite. As it represents one of the largest real-world datasets represented in RDF and OWL, managing UniProt it is often used as benchmark for scalability and reasoning capabilities of semantic repositories. While in the summer of 2007 the size of the RDF representation of UNIPROT was about 384M statements, in the beginning of 2008 its size approaches a billion of explicit statements. The RDF graph defined in the UNIPROT ontology is highly interconnected, which has significant impact on the reasoning speed of semantic repositories.

Over the last year the vision for the Semantic Web as a web of structured data started materializing in the form of the so-called “linked data”: an initiative for publishing RDF data on the web, following some principles which allow exploration/navigation and interlinking. The Linking Open Data project⁶ (LOD) has already collected over 40 datasets which are interlinked between each other and contain in total above 3 billion RDF statements. The central dataset within LOD is DBPedia⁷ - an RDF dataset derived through extraction of structured information from Wikipedia⁸. It currently contains about 218 million explicit statements describing and interconnecting about 2 million entities. DBPedia is also very well interconnected with several other datasets, e.g. W3C's representation of Wordnet in RDF(S)/OWL. As in the case of UNIPROT, it is not a benchmark, but rather a challenging non-synthetic datasets which could be used for benchmarking purposes.

3.3. Full-Cycle Benchmarking

An important quality of the benchmark is to cover a complete scenario of usage of the functionality of the tool. For instance, passing LUBM involves the following activities of a SR:

- Parsing of RDF/XML;
- Storage/persistence;
- Indexing;
- Reasoning;
- Query evaluation.

The performance is measured for loading and for query evaluation – two tasks which provide indication about the performance regarding all the activities and thus provide a consistent picture about the efficiency of specific tool or approach. Without such complete picture, there is no empirical proof for the performance regarding some of the activities. For instance, inference can take place both at load or at query time or to be split into different phases between those two modalities of usage. There is no perfect inference schema or configuration for SR – different approaches provide optimal results in different situations; forward-chaining on top of very dynamic dataset with large “inferred closure” is suboptimal. The same applies to indexing – loading can be very efficient if little or no indices are created and stored; this however will have impact on the query performance.

When evaluating the performance of a SR or a specific configuration, one should pay most attention to the results from full-cycle-benchmarks, which allow consistent interpretation of its advantages and the disadvantages. If for instance, only the results of loading an LUBM dataset are presented, without query

⁵ www.uniprot.org/

⁶ <http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

⁷ <http://wiki.dbpedia.org/>

⁸ <http://www.wikipedia.org/>, the most popular open web encyclopaedia.



FP7 - 215535

Deliverable 5.5.1

performance evaluation, there is no trustworthy indication about the complexity of the inference and indexing involved in this task.



4. State of the Art of Scalable Reasoning

I will try to provide below the most relevant evaluation data publicly available about the state of the art of “classical” semantic repositories. Before analyzing evaluation data, it should be noted that there are very few full-cycle-benchmarking results reported in the range above one hundred millions of statements.

Based on all public results, currently the only schema for implementation of a semantic repository, which allows for sound and complete reasoning and query evaluation on top of a billion of RDF triples, is total materialization (see section 2.1). The UNIPROT results of RDF Gateway, [23], declared to be a deductive database, can be considered an exception. However, there is very limited information available about the loading (takes “several days”, without inference) and query evaluation, especially when the query requires non-trivial inference.

I will first comment on some scalability results related to OWL DL and then move towards the more elaborate discussion regarding specific semantic repositories, which can handle huge datasets with reasoning of smaller complexity.

4.1. Scalability of DL Reasoning

I am not aware of published results about practically usable setup of a semantic repository endorsing OWL-DL semantics on top of more than 10 million statements⁹. As mentioned in section 2.2, the fundamental reason is that the DL formalisms (including OWL Lite) are not tractable. As presented in the following subsections, DL results on larger datasets are reported under specific constraints:

- incomplete or approximate reasoning;
- support for concrete reasoning tasks, but not the full functionality expected by a semantic repository. Such examples are specific modes of inconsistency checking.

Below, I provide references to some of the outstanding results in this area, although they do not match the principle scalability targets set out in section 1. The purpose is to provide evidence about the current capabilities of the engines working with semantics at this class of complexity, so that they can be used as basis for outlining the corresponding targets for incomplete reasoning.

UOBM Benchmark Results

I have summarized the most interesting UOBM benchmark (see section 3.2) results in Table 1. The size of the respective datasets is indicated in millions of explicit statements. The information about Lite-1 and DL-1 is given in order to provide evidence how loading and query times depend on the size of the UOBM dataset for the different tools. QTPR is a metric defined in section 3.

Table 1. Results for UOBM (Simple DL Reasoning)

<i>OW=SwiftOWLIM</i> <i>MI=MINERVA</i>	Lite-1 (0.2 MSt.)		DL-1 (0.2 MSt.)		Lite-10 (2.1 MSt.)		DL-10 (2.2 MSt.)		DL-20 (4.4 MSt.)
	OW	MI	OW	MI	OW	MI	OW	MI	OW
Loading time (sec)	10	950	12	1000	94	13 000	161	9 500	446
Loading speed (st./s.)	20 000	211	16 667	200	22 340	162	13 665	232	9 865
Aver. Query Time (s.)	0.054	0.1	0.049	0.15	2.452	1	1.686	0.5	7.967
Aver. QTPR (msec.)	1.2		0.3		4.8		3.3		6.9

⁹ An exception are the results of SHER, discussed later on in this section. However, it is unclear to what extent they can be reproduced in a standard semantic repository benchmark.



Interpreting the above results, one should consider that the results for MINERVA, [25], are measured in a setup (Pentium IV 2.66 GHz, JDK 1.4.2), which is about twice slower than the one of OWLIM (Pentium Core 2 Duo 3.00 GHz, JDK 1.6).

According to [25], MINERVA was able to load the DL-10 set and answer the first 12 queries completely, and queries 13th and 14th incompletely. As no concrete query evaluation times are provided, the numbers that I provide in the table are guesses based on the figures.

SwiftOWLIM (ver. 2.9.1, [30]) is able to load DL-10 on a desktop machine (Java heap maximum set to 512MB) and answer completely all the queries, except the 15th, which require ABox reasoning not supported by SwiftOWLIM. Using an excerpt of the dataset published at IBM SHER reasoner's page¹⁰, SwiftOWLIM scaled up to DL-20 and DL-25 in 32-bit environment. The loading speed was decreasing in sub-logarithmic fashion. Some peculiarities in this dataset caused strange slowdowns when the complete DL-30 set is loaded. It seems to me that the UOBM dataset generator is far less predictable and transparent, compared to the one of the original LUBM.

In theory, for UOBM SwiftOWLIM should have been able to perform better in its “transitive” mode, which implements a hybrid reasoning strategy. In particular, the inferred closures of transitive, inverse, and symmetric properties are not materialized, but rather handled programmatically, in a sort of backward-chaining fashion. As reported in [31], this hybrid strategy was proven to be very efficient in handling purely synthetic datasets, as the one of the EXQUANT test defined in [40]. Out of a sudden, this was not the case with UOBM – SwiftOWLIM handles it more efficiently in its standard “total materialization” mode.

Follows my analysis on the UOBM results above:

- The complexity of incomplete reasoning against some light-weight DL ontologies and datasets can grow in linear dependency on the size of the dataset. The loading speed decreases in sub-logarithmic fashion. The query evaluation times grow fast, but QTPR grows linearly, i.e. the major reason for the lower query times are the growing result sets;
- It is important to note that the query times are far from the constant behavior expected by a RDBMS. The explanations are different for the different reasoning strategies. In the case of SwiftOWLIM, the queries are slowed down by the partial backward-chaining performed at query/retrieval-time.

Based on simple extrapolation, I believe that a “regularly” grown DL-30, can be loaded on 32-bit Java machine (e.g. less than 1.6 GB of Java heap). Experimental data show that the speed for loading these approx. 7M statements with DL semantics will be close to 8000 statement/sec. In terms of query evaluation performance, I expect QTPR values in the range 10-20 msec. On a 64-bit server hardware (given 15 GB of Java heap) SwiftOWLIM should be able to handle an UOBM DL-250 dataset, with size between 50 and 60 million explicit statements. The query evaluation performance can be expected to be in the range of 100 msec. QTPR. Such an experiment has not been carried out because the instance data generator for UOBM is not publicly available.

Recently, I got informal notice from the SHER team (see next section) that SwiftOWLIM successfully loaded a modified (larger and more complex) version of the DL-150 dataset, containing 45 million triples. The experiment took place on a 64-bit Linux server with 16GB of RAM; unfortunately the actual loading times were not recorded. This is an indication that the extrapolations that I made above regarding the scalability of SwiftOWLIM under UOBM are correct. I was also informed that BigOWLIM 3.0 was also successful in loading DL-150. More details about the setup can be found in see [8] and [9].

¹⁰ http://domino.research.ibm.com/comm/research_projects.nsf/pages/iaa_index.html (as of 26 Nov); [http://domino.research.ibm.com/comm/research_projects.nsf/pages/iaa_index.html/\\$FILE/dataset.zip](http://domino.research.ibm.com/comm/research_projects.nsf/pages/iaa_index.html/$FILE/dataset.zip) is the link for the dataset



SHER and ABox Summarization

[7] reports interesting results on the UOBM performance of SHER – a scalable high-performance reasoner developed at IBM. SHER’s scalability is related to its reasoning strategy, which can be outlined as follows:

1. The ABox is “summarized” and the resulting smaller ABox, together with the original TBox (the concept definitions), is populated in Pellet ([3], one of the outstanding “classical” DL reasoners);
2. SHER transforms the reasoning operations into inconsistency checking, that is performed by Pellet. For example, the instance retrieval query “give me all instances of concept C” can be modeled by adding to the ABox assertion “NOT x:C”, i.e. “there is no individual which belongs to C” and then checking consistency of the KB;
3. In case the summarized ABox appears consistent, then the original is also consistent, so, the according conclusions are drawn by SHER and the reasoning tasks ends;
4. If the summarized ABox appears inconsistent, SHER starts a procedure for “refinement” of the summary, and then the extended summary ABox is again checked through Pellet;
5. If the refined ABox appears consistent, SHER continues as in step 3 above. Otherwise it goes back to step 4 and continues with the refinements, i.e. with the “de-compression” of the summarized ABox, until all expansions relevant to the current task are made;
6. In case that no consistent ABox was found, SHER draws “justifications” for the inconsistency, i.e. sets of triples which render the ABox inconsistent. In the case that the original task was a particular query, these justifications represent the answers of the query. Otherwise, the justifications can be considered just as such, so, they could suggest noise or inconsistencies in the ABox.

The above schema allows scaling up particular types of inference tasks on DL ontologies considerably, as DL reasoning should be performed against the summarized ABox, which usually appears to be some orders of magnitude smaller than the original one.

[7] reports on the performance of instance retrieval queries for all 112 concepts of UOBM against the DL-30 dataset (approx. 7 million statements). The average query time measured is 35 sec. Considering that there are approx. 709 159 type assertions, it suggests QTPR in the range of 5 msec. This is a very impressive result for complete reasoning on top of such a large ABox. However, the following limitations of the approach should be considered:

- The instance retrieval queries are simpler than the ones included in UOBM.
- The approach of query answering through inconsistency checking is expensive for application on conjunctive queries. This could be an explanation of why [7] provides no results of evaluation with the original UOBM queries.
- Loading time is not reported.

[35] reports on a case study where SHER was used to match patient records to clinical trials using the SNOMED ontology. An important aspect of this experiment was that it involves a large TBox. To handle this, SHER implements another important optimization – it excludes from the process the irrelevant part of the TBox, namely, those concepts which are not referred in the ABox and which are not related to concepts that are referred there. This operation is performed through FACT++ ([21], another outstanding DL reasoner). Originally, there were about 500 thousand assertions in the TBox, out of which only 22 561 thousand remained and were used in the course of the ABox reasoning. The dataset put on test contained approx. 60 million statements – data from clinical records of 240 thousand patients for 1 year. The results can be summarized as follows:

- Loading and TBox “minimization” times are not reported;
- Queries took between 30 and 300 minutes and return between 2 and 5555 results. The QTPR ranges between 1 and 1830 seconds, with 213 sec. average.



Recently the SHER team at IBM conducted experiments at the scale of DL-100 and DL-150, [8] and [9]. The subject of evaluation is a hybrid approach which allows incomplete reasoning to be used to optimize the refinement strategy presented above. The results are interesting in two aspects:

- They prove that the approach of SHER scales up to 45 million assertions in UOBM extension with SHIN complexity.
- They demonstrate that the hybrid approach can deliver considerable improvements of the query evaluation performance, lowering the average query time from 100 to 15 minutes.

The results of SHER are interesting as they provide important evidence for a new approach of scaling up DL reasoning. The hardware used in the experiments is comparable to the one discussed below for the highly scalable repositories.

The approach of SHER can be compared with the incomplete reasoning scheme proposed in the LARKC project. In particular, the summarization step can be considered as a sort of abstraction, while the iterative refinement represents a kind of retrieval of an increased sample.



4.2. Systems and Datasets

I will focus on the following performance results of some of the state-of-the-art semantic repositories:

- LUBM benchmarks of AllegroGraph, BigOWLIM, Virtuoso, ORACLE, and DAML DB;
- UNIPROT loading of AllegroGraph and BigOWLIM.

Table 2 presents a summary of the tools and the hardware configurations behind the reported results.

Table 2. Systems and Configurations

Tool	Version	Configuration
AllegroGraph	AllegroGraph 64-bit RDFStore, ver. 3.0	Server 1xOpteron 844 (1.8 GHz) CPUs, 16GB RAM, 2xSATA
		Server 4xOpteron 844 (1.8 GHz) CPUs, 16GB RAM, 2xSATA
OWLIM	BigOWLIM 2.0 (ex. 0.9.x) BigOWLIM 3.0	Server 2xOpteron 270 (2GHz,dc) CPUs, 16GB RAM, 4xSATA
		Server 2xXeon 5420 (2.5GHz,qc) CPUs, 32GB RAM, 8xSAS
Virtuoso	Open-source version 5.5	Server 2xXeon 5335 (2GHz) CPUs, 16GB RAM, 6xSATA drives
ORACLE	ORACLE 11g	PC, single CPU (3GHz), 4GB of RAM, two SATA drives
DAML DB	Ver. 2.2.1.2, via Sesame ver. 1.2.6	Server 2xXeon E5345 CPUs (2.33 GHz), 16GB RAM, 6 TB storage (probably RAID over 10+ drives)

The equipment used for the runs of the different systems seem comparable – all the results report on hardware in the range from “several PCs with Gigabit connection” to inexpensive database server configurations. More details are provided in the sub-sections dedicated to the specific tools.

In general, it is important to note that the reader should consider the data below as indicative figures, which provide evidence about the current state of the art in scalable reasoning, rather than as comparative analysis between the tools. In the following sub-sections we first present the systems reviewed and next comment on various performance and scalability issues.

AllegroGraph

AllegroGraph RDFStore¹¹ is an RDF database with support for “SPARQL, RDFS++, and Prolog reasoning from Java applications”. In addition to the “basic” DBMS functionality it offers specific support for geo-spatial data handling and social network analysis.

AllegroGraph is a quad-store which does not perform reasoning during loading. The so-called RDFS++ reasoning¹² can be switched on during query processing. The semantics supported this way is comparable to OWL Horst (see section 2.3). RDFS++ reasoning allows AllegroGraph to deliver correct results on LUBM(50), however the query evaluation times reported in [13] suggest that it is Allegro Graph is considerably slower than BigOWLIM. Query evaluation data for larger versions of LUBM are not reported. Communication with AllegroGraph developers confirmed the expectation that, because of the selected inference strategy, query evaluation is getting increasingly expensive with the growth of the repository and it is impractical at larger scale.

One of the major developments in its release 3.0 is the so-called “federation” which allows grouping multiple stores (running locally or on a remote machine) within a single virtual store. Federation has the potential to considerably speed up the loading process, as the work is effectively distributed across multiple stores. As

¹¹ <http://agraph.franz.com/allegrograph/>

¹² <http://agraph.franz.com/support/documentation/3.0/reasoner-tutorial.html>



reported in [12] and [14], loading LUBM(8000) on a 4-CPU machine with a virtual store combining four local stores is three times faster than performing the same load in a single, unified (i.e. non-federated) store. On the other hand [14] reports that query evaluation gets 2-3 times slower in the federated store. This configuration of AllegroGraph delivers the highest average speed of loading for LUBM(8000), across all systems and results analyzed in this report; one should bear in mind however, that the loading performed by AllegroGraph involves no reasoning at all (in contrast to BigOWLIM and ORACLE 11g).

Further [12] reports that *"On Amazon's EC2, AllegroGraph loaded and indexed 10 Billion Triples derived from 1 Billion Telecom CDRs (Call Detail Records) into 10 large instances, 4 parallel loads per instance, in 6.19 hours"*. While this is a very impressive result on its own I will leave it without comment and will not include it in the scalable reasoning table and diagram below, because:

- There is no information of reasoning taking place in relation to this load;
- The above quotation is the only information publicly available; there are no indications about the complexity of the dataset or any data about queries that were evaluated on top of this set.

BigOWLIM

OWLIM is a semantic repository implemented in Java and packaged as a Storage and Inference Layer (SAIL) for the Sesame¹³ RDF database. OWLIM is based on TRREE – a native RDF rule-entailment engine. The standard inference strategy is forward-chaining and materialization. The supported semantics can be configured through rule-set definition and selection. The most expressive pre-defined rule-set combines unconstrained RDFS and OWL Lite (see section 2.3). Custom rule-sets allow tuning for optimal performance and expressivity.

The two major varieties of OWLIM are SwiftOWLIM and BigOWLIM. In SwiftOWLIM, reasoning and query evaluation are performed in-memory, while, at the same time, a reliable persistence strategy assures data preservation, consistency, and integrity. BigOWLIM is the “enterprise” variety that deals with data and indices directly from disc or other file storage, which allows it to scale much better. Further, BigOWLIM’s indices are specially designed to allow efficient query evaluation against huge volumes of data. SwiftOWLIM can manage millions of explicit statements on desktop hardware. Given an entry-level server, BigOWLIM can handle billions of statements and serve multiple simultaneous user sessions. BigOWLIM 2.0 (ex-versioned 0.9.x) is a classical triple store. BigOWLIM 3.0 is a next version of the engine, which delivers the following improvements:

- Based on BigTRREE 3.0, supporting named graphs and triple sets (see section 2.2);
- It is compliant with SPARQL, through Sesame 2.0;
- Its persistency and indexing strategy, allow for smoother and more efficient scalability in the range of billions of triples.

Results of benchmarking of various versions of OWLIM can be found in [30]. We shall mention that BigOWLIM 3.0 is the system which demonstrates the most scalable reasoning results – the LUBM(20 000) run on 2.8 billion explicit statements. After materialization, the repository contained about 4.5 billion statements; the server used for this run is relatively powerful, but still meets the cost constraints set out in section 5.

However, the results of this run also do not comply with the “full-cycle benchmarking”, because we failed to collect query evaluation results, due to overflow caused by a huge query result. Such information will be

¹³ <http://www.openrdf.org/> - one of the most popular RDF database frameworks, developed by Aduna b.v. Two of the three systems, namely OWLIM and DAML DB, which were empirically proven to perform inference against hundreds of millions of statements, use Sesame as a framework which provide basic RDF infrastructure (i.e. RDF and query language parsers).



made available shortly. Considering the facts and the available data, the largest “full-cycle benchmarking” results are those from BigOWLIM 2.0 passing LUBM(8 000).

DAML DB

The server used for the DAML DB runs, see [36], seems more powerful than the others referred here. In terms of CPU power, the machine should be considered comparable to the others, since, to the best of my knowledge, the engines included and the benchmark setup benefit little from more than two CPU cores. Although there is no concrete information in [36], the storage system is probably much faster than the ones used in the runs that we discuss below.

In addition, [36] provides no concrete measurement data, but only diagrams where load and query times are presented on a logarithmic scale, with respect to the growth of the repository size. In order to be able to position it properly on my diagrams below, I made imprecise approximations for the figures behind the diagrams. I would be happy to update this document with more concrete data, shall such become available to me.

ORACLE

ORACLE offers RDF support as part of the Spatial option of its DBMS since version 10g R2. In version 11g this support is improved in several ways, among which:

- Support for the so-called OWL Prime dialect, [41], which is comparable with the owl-max semantics of OWLIM (see section 2.3). The Pellet DL reasoner is integrated for T-Box (schema level) inference.
- The efficiency of RDF loading and inference is considerably improved.

ORACLE supports RDF models with named graphs, i.e. it can be seen as a quadruple store. The semantics to be used for inference is defined in terms of the so-called rulebases, which essentially represent sets of entailment rules. Inference is implemented in terms forward-chaining and materialization – the results are stored in the so-called rule-indices. The initiative regarding inference is given to the user; she should take care to explicitly:

- Force inference, i.e. generation of the rule indices;
- Invalidate the rule indices when necessary, i.e. after statements are deleted.

Results from benchmarking ORACLE 11g are published in [32] and [33]. One should consider summing up the times for the several steps. For instance, the times reported for LUBM(8000) in [33] are as follows:

- Bulk-load: 30 h. 43 min;
- Loading into staging table via SQL*Loader: 2 h. 35 min;
- Inference (OWLPrime + Pellet on TBox; 4GB of RAM): 56.7 hours.

I shall acknowledge two important circumstances:

- the LUBM(8000) results for ORACLE are measured on single desktop computer; all the other results are measured on servers;
- ORACLE is also most “economic” with respect to the RAM usage – the above results reflect the inference run with 4GB of RAM, but results measured on 2GB systems, suggest graceful degradation of the performance when less memory is available.

Unfortunately, no query evaluation results are presented, i.e. there are no public data for full-circle benchmarking. Thus, there is no empirical proof of the inference capabilities and correctness and there is no indication about the query evaluation performance of ORACLE 11g.



Virtuoso

OpenLink Virtuoso is a “universal server” offering diverse data and metadata management facilities, e.g. XML management, RDBMS integration, full-text indexing, etc. The core engine of Virtuoso is an RDF database. The experiments of benchmarking Virtuoso 5.5 under LUBM are presented in [10]. Virtuoso 5.5 is a “quadruple store”, i.e. it supports RDF with named graphs. Virtuoso 5.5 is “64-bit”, although from the public documentation it is not clear whether 64-bit node IDs are supported (see section 4.5).

Virtuoso provides no inference support integrated in the database engine. Inference can be implemented on top of Virtuoso in two ways:

- Entailment and materialization through SPARQL queries after loading is completed;
- Query expansion (re-writing) at query time.

The developers of Virtuoso report in [10] the results of various “distributions of labor” between materialization and query expansion. As expected, it appears that extensive materialization can save a lot of computations during querying, while the same query completeness is achieved.

In principle, this type of inference can be implemented on top of any DBMS – the complexity of the inference depends on the complexity of the query language supported. The principle concern here are that the semantics is not enforced by the engine, but it is rather a matter of hand crafted queries which may or may not correctly reflect the semantics of the ontology language primitives.

The data provided in [10] leaves several questions open:

- The queries of LUBM are modified, so, it is not clear: (i) how the query evaluation performance compares with this of other engines benchmarked with the original LUBM queries and (ii) whether the query results are truly complete and whether the implemented inference mechanisms are correct and sufficient for a full LUBM run.
- No performance data is provided for entailment and materialization. I.e. there are no indications about the performance and efficiency of the inference various inference configurations that were put on test.
- Implementing the semantics of transitive properties via query-based entailment and materialization, as presented in [10], requires recursive query evaluation. However, there are no comments on the implementation of such behaviour in Virtuoso.

Overall, the strengths of Virtuoso seem to be on efficient query evaluation and handling large volumes of relatively simple queries.

Version 6.0 is currently in development, one of the major advances there is the so-called “cluster mode” which allows distribution across several machines – preliminary information from OpenLink¹⁴ provides interesting insights about the design and the performance of the distributed version.

YARS2

Results related to indexing and querying 7 billion statements in cluster of 16 machines are published for YARS2, [19]. There is no information about the load time and no inference takes place. The query evaluation times indicate average values in the range of hundreds of milliseconds. However, they cover only simple queries with one or two joins. For these reasons, these experiments with YARS2 are not discussed below.

¹⁴ <http://www.openlinksw.com/dataspace/oerling/weblog/Oerling%20Erling%27s%20Blog/1335>



4.3. Loading Performance

Let me start with the loading performance. For the engines and experimental setups listed here, loading involves:

- Parsing of the input RDF;
- Persisting the data and indexing (including the implicit statements inferred);
- Inference, namely materialization through forward-chaining, for OWLIM, DAML DB, and ORACLE.

Table 3 presents the loading time and speed of the engines for the different datasets. For each run I provide the following data:

- **Scale and Time**, in number of explicit statements loaded and hours, respectively;
- **Speed**: average speed of loading, in thousands of statements per second;
- **Overall complexity index**: average of the Forward-Chaining and the Parsing and Indexing Complexity indices. It represent a combined measure for the complexity of the loading task;
- **Forward-chaining semantics**: what language is used for forward-chaining and materialization during the loading, if any;
- **Forward-chaining complexity-index**: my subjective estimate about the complexity of the reasoning during this run;
- **Parsing & Indexing Complexity**: my subjective estimate about the complexity of the loading tasks that are not related to indexing. It includes the specifics of the concrete run and system, e.g.: rich RDF model, full-text search indexing, etc.
- **Rich RDF Model**: whether a basic RDF data-model is supported or an extended one (e.g. such with support for named graphs; see section 2.2).

Table 3. Loading Performance

Semantic Repository	Dataset / Run	Scale (mill. ex.st.)	Speed (KSt./sec.)	Overall Compl. Index	Time (hours)	Forward-Chaining Semantics	Forw.Ch. Compl. Index	Pars.& Index. Compl.	Rich RDF Model
BigOWLIM 2.0	LUBM(8000,0)	1,068	8,725	14	34.00	OWL-Horst	20	7	-
BigOWLIM 2.0	LUBM(8000,0)	1,068	20,565	9	14.43	RDFS	10	7	-
BigOWLIM 2.0	LUBM(1000,0)	138	17,055	14	2.25	OWL-Horst	20	7	-
BigOWLIM 2.0	LUBM(1000,0)	138	34,743	9	1.10	RDFS	10	7	-
BigOWLIM 2.0	UNIPROT	383	4,572	29	23.27	OWL-Max	50	7	-
BigOWLIM 2.0	UNIPROT	383	14,263	16	7.46	RDFS+	25	7	-
BigOWLIM 3.0	LUBM(1000,0)	138	11,897	15	3.22	OWL-Horst	20	10	+
BigOWLIM 3.0	LUBM(8000,0)	1,068	12,816	15	23.15	OWL-Horst	20	10	+
BigOWLIM 3.0	LUBM(20000,0)	2,760	10,534	15	72.78	OWL-Horst	20	10	+
AllegroGraph 3.0 Fed.	LUBM(8000,0)	1,107	37,273	5	8.25	none	0	10	+
AllegroGraph 3.0 Fed.	UNIPROT	234	55,085	5	1.18	none	0	10	+
Openlink Virtuoso ver. 5.0	LUBM(8000,0)	1,068	36,900	5	8.04	none	0	10	+



One can see that the results for loading are comparable, taking into account that the engines differ in features. Taking BigOWLIM's results from LUBM(1000), one can observe how the difference in the semantics supported can alter the loading time almost by a factor of three.

4.4. Query Evaluation Performance

The second major performance criteria that I analyze here is query evaluation speed. There is very little public information available about query performance on datasets with close to one billion statements:

- The YARS2 data in [19] – they were excluded for the reasons given above;
- The results of BigOWLIM, evaluating the LUBM(8000) queries in [30].
- The results of DAML DB for the LUBM(8000), published in [36] – those are relevant, but they lack concrete numbers, rather provide graphic on a logarithmic scale instead.

In Table 4 I present results published in [30], which compare the query evaluation times of two versions of BigOWLIM: 0.9.5 and 0.9.6/2.0. It presents the number of results returned for each of the queries and the query evaluation times of each version for each of the queries. We can see that the number of the results returned varies between 83 millions and 4, i.e. there is a difference of some 7 orders of magnitude. Obviously, the criteria for fast evaluation between queries number 1 and 6 are different. For this purpose I adapt the QTPR (query-time-per-result) metric introduced in section 3.

Table 4. Query Evaluation Performance for LUBM(8000), 1 billion statements

Query No	Number of results	BigOWLIM 0.9.5		BigOWLIM 0.9.6/2.0	
		Time (ms)	QTPR	Time (ms)	QTPR
1	4	3 962	991	100	25
2	2 528	1 144 726	453	181 015	72
3	6	24 506	4 084	100	17
4	34	123 978	3 646	182	5
5	719	31 018	43	92	0
6	83 557 706	169 375	0	106 216	0
7	67	123 806	1 848	154	2
8	7 790	42 514	5	914	0
9	2 178 420	2 318 412	1	738 032	0
10	4	37 704	9 426	102	26
11	224	413	2	75	0
12	15	1 317	88	453	30
13	37 118	22 536	1	16 061	0
14	63 400 587	383 261	0	83 792	0
Average			1471		13

The average QTPR of BigOWLIM ver. 0.9.6/2.0 on LUBM for dataset of one billion statements is about 13 milliseconds. The result of the older version (0.9.5) of the same engine is over a hundred times lower. The major difference between the two versions is that the newer one makes better use of statistics for the sake of query plan optimizations, similar to those employed in the RDBMS. The difference in the results demonstrates an expected fact, namely, that query optimization is crucial for datasets of this size. As this type of query optimizations are practically impossible for a reasoning engine, which performs backward-chaining in the process of query evaluation, the above result provides evidence that such engines will face principle problems with their query evaluation performance on large datasets.



In my interpretation the query evaluation results given in [36] suggest the following query times of DAML DB over a 850 million statements subset of LUBM(8000):

- Query 1 time: ~1 hour; QTPR: 15 min. Obviously DAML DB's performance here can be a subject of improvement.
- Query 14 time: ~6 hours; QTPR ~0.3 msec. Although this is much slower than the 83 sec. needed by BigOWLIM, the QTPR is good;
- Query 9 time: ~1 hour; QTPR ~2 msec. Again, the QTPR is good.

Considering that the current version of BigOWLIM works with "simple" RDF model (e.g. without support for Named Graphs), one can expect that a well tuned engine can achieve at present speeds in the range of 10-30 milliseconds QTPR on LUBM(8000), depending on its other features.

4.5. 64-bit Scalability Notes

On the implementation side, one should consider a limitation that is clearly evident in Virtuoso and BigOWLIM – 32-bit node identifiers. In most of the platforms and programming languages nowadays the engineers have a choice between 32-bit and 64-bit integer values to be used as identifiers or references in the main memory. While handling, for instance, 40-bit integers is possible, it would often come at the same cost as dealing with 64-bits.

The architectures of most of the semantic repositories involve dealing with identifiers of arcs (statements) and nodes (URIs and literals) in the RDF graph. The number of the statements is larger than the number of the nodes; in a typical dataset, there are on average 3-5 explicit statements per node. Going beyond couple of billion statements requires 64-bit statement IDs – this is the case with BigOWLIM 3.0 and Virtuoso 6.0¹⁵; there are indirect indications that AllegroGraph is also not limited to 32-bit statement identifiers.

In BigOWLIM 3.0 32-bit integers are used as identifiers of the RDF nodes. This means, that there could be at most 4 billion nodes in the graph(s) handled by such repository, which implies a practical limitation of the size of the graph in the range of 10-20 billion explicit statements. Although such detailed technical information is not available from other vendors, on the basis of the publicly available data I believe that this constraint applies to most of the other repositories. While there are no technical or theoretical problems for building versions of the tools which deal with larger node identifier ranges, one should be aware that this would have impact on the performance of the tools as larger identifiers require more memory and larger data transfers.

4.6. Distributed Reasoning Notes

Few of the most scalable results presented today employ distributed, federated, or clustered repositories (e.g. those of AllegroGraph, YARS, and Virtuoso). While this is an obvious approach for achieving better scalability, a repository implemented under such schema is faced with several well known distribution issues researched in the database community over the last couple of decades. In summary, there is still not reasonably expensive approach for distribution of a database (the data and the indices) across several machines without very serious penalties in terms of query performance. An interesting discussion on the subject, with many historical references and comments can be found at [6]. An excellent overview of the various issues related to the distribution of an RDF repository can be found in [11].

As most of the reasoning tasks can either be seen as or be reduced to query answering, the same principle hurdles shall be addresses in a distributed semantic repository. There is also research in distributed reasoning in the KR community, most of it based on partitioning of ontologies or more generally knowledge bases: [15] and [38] presented approaches related to DL ontologies; more general overview on the subject can be found in [27].

¹⁵ <http://www.openlinksw.com/dataspace/oerling/weblog/Oerri%20Erling%27s%20Blog/1335>



5. Performance Targets

This section introduces measurable performance targets for semantics repositories, based on the discussion on the relevant tasks and factors (section 3), and on the analysis of the current state of these systems (section 4). My aim is to provide measuring sticks, which could facilitate the development of this type of technology over the next couple of years.

I define four semantic repository performance targets (SRPT) as follows:

SRPT1: Complete LUBM reasoning. LUBM(8000) should be loaded and its queries shall be answered in sound and complete fashion. LUBM requires light-weight, tractable OWL reasoning. The dataset allows for easy partitioning into isolated sub-graphs.

Target: loading speed 100 000 explicit st./sec.; average QTPR below 4 msec.

SRPT2: Complete UNIPROT reasoning. The UNIPROT dataset should be loaded; the forward chaining complexity of UNIPROT is higher than this of LUBM. Queries comparable in complexity to the set published at RDF Gateway's UNIPROT experiments shall be evaluated; the queries shall involve the LIKE operator or similar.

Target: loading speed 20 000 explicit st./sec.; average QTPR below 300 msec.

SRPT3: Incomplete OWL DL ABox reasoning. A dataset corresponding to the DL version of the UOBM benchmark for 500 universities should be generated (approx. 110M st.) The corresponding queries should be answered in incomplete fashion while the following IR-style accuracy measures should be met: precision 95% at recall 25%.

Target: loading speed 50 000 explicit st./sec.; average QTPR below 30 msec.

SRPT4: Incomplete UNIPROT reasoning. As SRPT2, but with incomplete query answering; the following accuracy measures should be met: precision 98% at recall 10%.

Target: loading speed 50 000 explicit st./sec.; average QTPR below 100 msec.

When using the above-mentioned performance targets, the following should be considered:

- The targets shall be met on a single “general purpose” server with cost up to 10 000 EURO by the end of 2010. Moor's law is likely to stultify them in a longer term;
- Adjustments may be necessary if tests are made with version of UNIPROT which differs in size or complexity from the version available in the fall of 2007;
- Relevance ranking is not properly covered by the above specified targets.

I do not define specific targets for distributed semantic repositories. Since at present distributed implementations cannot beat the price/performance offered by single-server installations, the above goals should provide good test for distributed implementations. In a distributed set up, the price constraint should apply to the cost of the set of the machines, including the network/connectivity hardware.

The scalability targets defined here are just four points in the multidimensional space of the semantic repository performance. Thus, they do not define well-founded criteria for evaluation of such engines. They are rather meant as buoys that could help the navigation in the scalable reasoning area.



6. Conclusion

It is important to note that the goals of both projects which inspired this paper, namely LARKC and RASCALLI, are not to improve the performance of the “classical” reasoning engines. The primary target in LARKC is a new reasoning paradigm, a new definition of the tasks to be performed and objectives to be met by a semantic repository. In RASCALLI scalable reasoning is a secondary target and the objectives go in the direction of cognitive models far beyond the dangerous convenience of the classical mathematical logic.

Still, based on the state of the art analysis and the above-defined performance targets, Table 5 provides estimates about the expected performance and scalability advances.

Table 5. *Before and After*

	2007 SOA	2010 Targets
Maximum scale	1 billion statements	20 B st. (complete) 100 B st. (incomplete)
Speed of loading 1B statements	2 000-20 000 st./sec.	10 000-100 000 st./sec
Query evaluation against 1B statements (average QTPR)	10 msec.	4 msec. (complete) 1 msec. (incomplete)
Relevance ranking	Not present	Present
Inference strategy control	None/heavy	Plug-able inference modules Cost/benefit-based control

Extending my initial comments, it should be realized that there is limited utility in comparing the performance of the technology that LARKC aims at against the performance of today’s engines. I would make an analogy with a comparison between the advances made by the early aircrafts and sea ships, based on measurements with a lead.

An earlier version of this text has been included in the LARKC project documentation.

I want to thank professors Dieter Fensel and Frank van Harmelen for their comments and encouragements to publish this document, as well as Vassil Momtchev, Borislav Popov, and Uwe Keller for their corrections and suggestions. I shall express special gratitude to Zlatina Marinova, who took the burden of editing the manuscript, bringing it in a much more readable and consistent shape.

Finally, the semantic repositories landscape would have been a much different place without the work of Damyan Ognyanoff and Ruslan Velkov on OWLIM. I believe that without their efforts and engineering creativity, the overall level of ambition would have been much lower.



7. References

- [1] Brickley, D., Guha, R.V, eds. (2004). *Resource Description Framework (RDF) Schemas*. W3C Recommendation. <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>
- [2] Broekstra, J. (2005). *Storage, Querying and Inferencing for Semantic Web Languages*. Ph.D. Thesis, Vrije Universiteit Amsterdam, SIKS Dissertation Series No. 2005-09, ISBN 90 9019 2360. <http://www.cs.vu.nl/~jbroeks/#pub>
- [3] Clark & Parsia LLC. (2007). *Pellet open source OWL DL reasoner in Java*. <http://pellet.owldl.com/>
- [4] Carroll, J. J; Bizer, B; Hayes, P; Stickler, P. 2005. *Named Graphs, Provenance and Trust*. WWW2005, <http://www2005.org/cdrom/docs/p613.pdf>
- [5] Dean, M; Schreiber, G. – editors; Bechhofer, S; van Harmelen, F; Hendler, J; Horrocks, I.; McGuinness, D. L; Patel-Schneider, P. F.; Stein, L. A. (2004). *OWL Web Ontology Language Reference*. W3C Recommendation, 10 Feb. 2004. <http://www.w3.org/TR/owl-ref/>
- [6] DeWitt, D. J; Stonebraker, M. (2008). *MapReduce: A major step backwards*. *The Database Column*. January 17, 2008. <http://www.databasecolumn.com/2008/01/mapreduce-a-major-step-back.html>
- [7] Dolby, J; Fokoue, A; Kalyanpur, A; Kershenbaum, A; Ma, L; Schonberg, E; Srinivas, K. (IBM) (2007). *Scalable semantic retrieval through summarization and refinement*. In Proc. of the AAAI 2007 conference, July 2007, Vancouver, Canada. Page 299. [http://domino.research.ibm.com/comm/research_projects.nsf/pages/iaa.index.html/\\$FILE/SHER-AAAI2007.pdf](http://domino.research.ibm.com/comm/research_projects.nsf/pages/iaa.index.html/$FILE/SHER-AAAI2007.pdf)
- [8] Dolby, J; Fokoue, A; Kalyanpur, A; Ma, L; Patel, C; Schonberg, E; Srinivas, K; Sun, X. (IBM) (2008). *Efficient reasoning on large SHIN Aboxes in relational databases*. (not published yet).
- [9] Dolby, J; Fokoue, A; Kalyanpur, A; Ma, L; Patel, C; Schonberg, E; Srinivas, K; Sun, X. (IBM) (2008). *Scalable Conjunctive Query Evaluation Over Large and Expressive Knowledge Bases*. (not published yet).
- [10] Erling, O. (OpenLink Software). (2008). LUBM and Virtuoso. <http://virtuoso.openlinksw.com/wiki/main/Main/VOSArticleLUBMBenchmark>. 1 Feb, 2008.
- [11] Erling, O; Mikhailov, I. (OpenLink Software) (2008). *Towards Web-Scale RDF*. ISWC2008 submission. www.openlinksw.com/weblog/oerling/2008iswc_webscale_rdf.pdf
- [12] Franz Inc. (2008). *High-performance Storage and Querying*. <http://agraph.franz.com/allegrograph/> (as of 16 Jun, 2008).
- [13] Franz Inc. (2008). *AllegroGraph 3.0 Benchmark for LUBM-50-0*. http://agraph.franz.com/allegrograph/agraph_bench_lubm50.lhtml (as of 16 Jun, 2008).
- [14] Franz Inc./Garry King. (2008). *Querying Federated Knowledge for Web 3.0*. Presentation for a webinar given on 21st Feb, 2008. <http://agraph.franz.com/> ...
- [15] Grau, B; Parsia, B; Sirin, E; Kalyanpur, A. (2005). *Automatic Partitioning of OWL Ontologies Using E-Connections*. In Proc. of the 2005 Int. Workshop on Description Logics (DL-2005).
- [16] Grosz, B; Horrocks, I; Volz, R; Decker, St. (2003). *Description Logic Programs: Combining Logic Programs with Description Logic*. In Proc. of WWW2003, Budapest, May 2003.
- [17] Guo, Y; Pan, Z; and Heflin, J. (2004). *An Evaluation of Knowledge Base Systems for Large OWL Datasets*. *Journal of Web Semantics*, 3(2), 2005, pp. 158-182. <http://www.websemanticsjournal.org/ps/pub/2005-16>



- [18] Fokoue, A; Kershenbaum, A; Ma, L; Schonberg, E; Srinivas, K. (IBM) (2006). *The Summary Abox: Cutting Ontologies Down to Size*. In: Proc. ISWC-06. Volume 4273 of LNCS.
- [19] Harth, A; Umbrich, J; Hogan, A; Decker, S. (2007) *YARS2: A Federated Repository for Querying Graph Structured Data from the Web*. In Proc. ISWC 2007. <http://sw.deri.org/2007/02/swsepaper/iswc2007.pdf>
- [20] Hayes, P. (2004). *RDF Semantics*. W3C Recommendation 10 Feb. 2004. <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>
- [21] Horrocks, I. (1998). *Using an expressive descriptive logic: fact or fiction?* In proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98), pp. 636–647.
- [22] Horrocks, I., Patel-Schneider, P. F., Bechhofer, S., Tsarkov, D. (2005) *OWL Rules: A Proposal and Prototype Implementation*. [Journal of Web Semantics 3 \(2005\), pp. 23-40.](http://www.w3.org/TR/owl-features/)
- [23] Intellidimension, Inc. (2005). *Experiments with Uniprot RDF and RDF Gateway*. <http://labs.intellidimension.com/uniprot/> (as of Nov 4, 2007; Last updated Jun 2005).
- [24] Klyne, G; Carrol, J. J; (eds). (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation 10 Feb. 2004. <http://www.w3.org/TR/rdf-concepts/>
- [25] Ma, L; Yang, Y; Qiu, Z; Xie, G; Pan, Y. (2006) *Towards A Complete OWL Ontology Benchmark*. In Proc. of the 3rd European Semantic Web Conference (ESWC 2006). Budva (Montenegro).
- [26] Momtchev, V.; Kiryakov, A.: *ORDI Second Generation Specification*. October 2006, http://ordi.sourceforge.net/docs/ORDI_SG_Specification.pdf
- [27] MacCartney, B; McIlraith S. A; Amir, E; Uribe, T. (2003). *Practical partition-based theorem proving for large knowledge bases*. In Proc. of IJCAI 2003.
- [28] MacGregor, R; Ko, I.-Y. (2003). *Representing Contextualized Data using Semantic Web Tools*. Proc. of the 1st International Workshop on Practical and Scalable Semantic Systems; available at <http://km.aifb.uni-karlsruhe.de/ws/psss03/proceedings/macgregor-et-al.pdf>
- [29] Motik, B; Sattler, U; Studer, R. (2005) *Query Answering for OWL-DL with Rules*. [Journal of Web Semantics 3 \(2005\), pp. 41-60.](http://www.w3.org/TR/owl-features/)
- [30] Ontotext Lab. (2008) *OWLIM – Pragmatic OWL Semantic Repository*. Presentation. <http://www.ontotext.com/owlim/OWLIMPres.pdf> (as of 18 Jun, 2008).
- [31] Ontotext Lab. *SwiftOWLIM: System Documentation*. Version 2.9.1 of 10th of September 2007. <http://www.ontotext.com/owlim/v2.9.1/OWLIMSysDoc.pdf>
- [32] Oracle Corp. (2008). *Oracle Semantic Technologies Inference Best Practices with RDFS/OWL*. An Oracle White Paper. February 2008.
- [33] Oracle Corp. (2008). *Semantic Technologies Product Performance*. http://www.oracle.com/technology/tech/semantic_technologies/htdocs/performance.html, as of 18th of June, 2008.
- [34] Prud'hommeaux, E; Seaborne, A. (2008). *SPARQL Query Language for RDF*. W3C Recommendation 15 January 2008. <http://www.w3.org/TR/rdf-sparql-query/>
- [35] Schonberg, E; Srinivas, K; Kalyanpur, A; Cimino, J; Patel, C; Dolby, J; Kershenbaum, A; Ma, L; Fokoue, A. (2007). *Matching Patient Records to Clinical Trials Using Ontologies*. In Proc. ISWC 2007
- [36] Ruhloff, K; Dean, M; Emmons, I; Ryder, D; Sumner, J. (2007). *An Evaluation of Triple-Store Technologies for Large Data Stores*. In Proc. of Scalable Semantic Systems Workshop (SSSW 2007).



- [37] ter Horst, H. J. (2005) *Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity*. In Proc. of ISWC 2005, Galway, Ireland, November 6-10, 2005. LNCS 3729, pp. 668-684.
- [38] Schiaffino, R; Fokoue, A; Kalyanpur, A; Kershenbaum, A; Ma, L; Patel, C; Schonberg, E; Srinivas, K. (2006). *Computing OWL Ontology Decompositions Using Resolution*. The Second International Workshop on Modular Ontologies.
- [39] Swiss Institute of Bioinformatics. (2007). *UniProt RDF*. <http://dev.isb-sib.ch/projects/uniprot-rdf/>
- [40] Weithöner, T., Liebig, T., Luther, M., Böhm, S. (2006). *What's Wrong with OWL Benchmarks? SSWS2006*. <http://www.cs.vu.nl/~holger/ssws2006/Proceedings-SSWS06.pdf>
- [41] Wu, Alan / ORACLE. (2007). *A Scalable RDBMS-Based Inference Engine for RDFS/OWL*. Invited talk in front of the "Database and Ontologies" group; 10 Oct, 2007; <http://ontolog.cim3.net/cgi-bin/wiki.pl?DatabaseAndOntology>. http://ontolog.cim3.net/file/work/DatabaseAndOntology/2007-10-18_AlanWu/RDBMS-RDFS-OWL-InferenceEngine--AlanWu_20071018-alt.ppt