



## LarKC

*The Large Knowledge Collider*

*a platform for large scale integrated reasoning and Web-search*

FP7 – 215535

---

# D4.7.1 Initial Evaluation and Revision of Plug-ins Deployed in Use-cases

---

**Coordinator: Zhisheng Huang (VUA)**

**With contributions from: Zhisheng Huang (VUA), Barry Bishop (STI Innsbruck), Florian Fischer (STI Innsbruck), Frank van Harmelen (VUA), Jose Quesada (MPG), Lael Schooler (MPG), Gaston Tagni (VUA), Annette ten Teije (VUA), Axel Tenschert (HLRS), Alexey Cheptsov (HLRS), Emanuele Della Valle (CEFRIEL), Vassil Momtchev (OntoText), Yi Zeng (WICI), Yan Wang (WICI), Ning Zhong (WICI)**

**Quality Assessor: Mick Kerrigan (STI Innsbruck)**

**Quality Controller: Frank van Harmelen (VUA)**

Document Identifier:	LarKC/2008/D4.7.1/V1.0
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	version 1.0.0
Date:	September 30, 2009
State:	final
Distribution:	public



## EXECUTIVE SUMMARY

The essence of the LarkC project is to go beyond notions of absolute correctness and completeness in reasoning. We are looking for retrieval methods that provide useful responses at a feasible cost of information acquisition and processing. Therefore, generic inference methods need to be extended to non-standard approaches. In consequence, traditional metrics such as completeness and correctness for reasoning need to be replaced by metrics that ratio the utility of solutions with their related costs as a means to evaluate the chosen problem solver. Furthermore, as reasoners can employ multiple techniques and implement different strategies a framework for evaluating and measuring the quality and performance of them must exist that is able to take into account these different strategies. In this document, we develop an initial framework of evaluation and benchmarking of reasoners deployed within the LarkC platform. We define the evaluation methods, measures, benchmarks, and performance targets for the plug-ins to be developed in various tasks such as approximate reasoning with interleaved reasoning and selection and rule-based reasoning. These are examined with the perspectives of the use-cases of WP6 and WP7, as well as in terms of synthetic benchmarks. This document also discusses some initial ideas and approaches for evaluating parallel and distributed reasoning, rule-based reasoners, granular reasoning and, provides some ideas about the evaluation from a psychological and cognitive perspective.



## DOCUMENT INFORMATION

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

<b>Deliverable</b>	<b>Number</b>	4.7.1	<b>Title</b>	Initial Evaluation and Revision of Plug-ins Deployed in Use-cases
<b>Work Package</b>	<b>Number</b>	4	<b>Title</b>	Reasoning and Deciding

<b>Date of Delivery</b>	<b>Contractual</b>	M18	<b>Actual</b>	30-Sept-09
<b>Status</b>	version 1.0.0		final <input checked="" type="checkbox"/>	
<b>Nature</b>	prototype <input type="checkbox"/> report <input checked="" type="checkbox"/> dissemination <input type="checkbox"/>			
<b>Dissemination Level</b>	public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>			





<b>Authors (Partner)</b>	Zhisheng Huang (VUA), Barry Bishop(STI Innsbruck), Florian Fischer (STI Innsbruck), Frank van Harmelen (VUA), Jose Quesada (MPG), Lael Schooler (MPG), Gaston Tagni (VUA), Annette ten Teije (VUA), Axel Tenschert (HLRS), Alexey Cheptsov (HLRS), Emanuele Della Valle (CEFRIEL), Vasil Momtchev (OntoText), Yi Zeng (WICI), Yan Wang (WICI), Ning Zhong (WICI)			
<b>Resp. Author</b>	Zhisheng Huang (VUA)		<b>E-mail</b>	huang@cs.vu.nl
	<b>Partner</b>	UIBK, WICI, MPG, CEFRIEL, OntoText, HLRS	<b>Phone</b>	+31 (20) 5987823

<b>Abstract (for dissemination)</b>	<p>The essence of the LarKC project is to go beyond notions of absolute correctness and completeness in reasoning. We are looking for retrieval methods that provide useful responses at a feasible cost of information acquisition and processing. Therefore, generic inference methods need to be extended to non-standard approaches. In consequence, traditional metrics such as completeness and correctness for reasoning need to be replaced by metrics that ratio the utility of solutions with their related costs as a means to evaluate the chosen problem solver. Furthermore, as reasoners can employ multiple techniques and implement different strategies a framework for evaluating and measuring the quality and performance of them must exist that is able to take into account these different strategies. In this document, we develop an initial framework of evaluation and benchmarking of reasoners deployed within the LarKC platform. We define the evaluation methods, measures, benchmarks, and performance targets for the plug-ins to be developed in various tasks such as approximate reasoning with interleaved reasoning and selection and rule-based reasoning. These are examined with the perspectives of the use-cases of WP6 and WP7, as well as in terms of synthetic benchmarks. This document also discusses some initial ideas and approaches for evaluating parallel and distributed reasoning, rule-based reasoners, granular reasoning and, provides some ideas about the evaluation from a psychological and cognitive perspective.</p>
<b>Keywords</b>	Reasoning, Evaluation, Benchmarking, Web Scale Reasoning





## 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 Email: 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 RE- SPONSABILITA LIMITATA Milano, Italy Email: emanuele.dellavalle@cefriel.it
CYCROP, RAZISKOVANJE IN EKSPERI- MENTALNI RAZVOJ D.O.O.		Michael Witbrock CYCROP, RAZISKOVANJE IN EKSPERI- MENTALNI 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 AD		Atanas Kiryakov, Ontotext Lab, Sofia, Bulgaria Email: naso@ontotext.com
SALTLUX INC.		Kono Kim SALTLUX INC Seoul, Korea Email: kono@saltlux.com
SIEMENS AKTIENGESELLSCHAFT		Dr. Volker Tresp SIEMENS AKTIENGESELLSCHAFT Muenchen, Germany Email: 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
VRIJE UNIVERSITEIT AMSTERDAM		Prof. Dr. Frank van Harmelen, VRIJE UNIVERSITEIT AMSTERDAM Amsterdam, Netherlands Email: Frank.van.Harmelen@cs.vu.nl
THE INTERNATIONAL WIC INSTI- TUTE, BEIJING UNIVERSITY OF TECHNOLOGY		Prof. Dr. Ning Zhong, THE INTERNATIONAL WIC INSTITUTE Mabeshi, Japan Email: zhong@maebashi-it.ac.jp
INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER		Dr. Paul Brennan, INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER Lyon, France Email: brennan@iarc.fr
INFORMATION RETRIEVAL FACILITY		Dr. John Tait, Dr. Paul Brennan, INFORMATION RETRIEVAL FACILITY Vienna, Austria Email: john.tait@ir-facility.org



# TABLE OF CONTENTS

LIST OF FIGURES	8
LIST OF ACRONYMS	9
1 INTRODUCTION	10
2 LARKC PLATFORM	12
2.1 LarKC Architecture . . . . .	12
2.2 LarKC reasoning plug-ins . . . . .	13
2.2.1 Reasoning APIs . . . . .	13
2.2.2 Reasoning Plug-ins . . . . .	14
3 A FRAMEWORK OF EVALUATION AND BENCHMARKING FOR ONTOLOGY REASONERS	15
3.1 General Consideration . . . . .	15
3.2 Goals and Criteria for Evaluation and Benchmarking of Reasoner Plug-ins	15
3.3 Measuring the Quality of Query Answers . . . . .	16
3.4 Workflows of Evaluation and Benchmarking . . . . .	17
3.5 A Specification Language for Golden Standards . . . . .	17
4 EVALUATION FROM A WP6 PERSPECTIVE	22
4.1 Alpha Urban LarKC . . . . .	22
4.2 Workflows of Alpha Urban LarKC . . . . .	23
4.3 Use Cases for Reasoning Plug-ins . . . . .	24
4.3.1 Standard RDFS/DL Reasoners . . . . .	24
4.3.2 Approximation Reasoners . . . . .	24
4.3.3 Rule-based Reasoners . . . . .	24
4.3.4 Parallel and Distributed Reasoners . . . . .	24
5 EVALUATION FROM A WP7A PERSPECTIVE	26
5.1 Linked Life Data . . . . .	26
5.2 Semantic Data Integration Workflow . . . . .	26
5.3 Evaluation criteria . . . . .	28
5.4 Use Case of Reasoning Plug-ins . . . . .	29
5.4.1 Description Logic Reasoners . . . . .	29
5.4.2 Rule-based Reasoners . . . . .	29
5.4.3 Approximation Reasoners . . . . .	30
5.4.4 Parallel and Distributed Reasoners . . . . .	30
6 EVALUATION OF APPROXIMATION REASONERS	31
6.1 Introduction and Motivation . . . . .	31
6.2 Multiple Forms to Approximate Answers . . . . .	32
6.3 Approximate Reasoning on the Semantic Web . . . . .	33
6.4 Further Studies . . . . .	33



7	EVALUATION OF RULE-BASED REASONERS	35
7.1	Synthetic Benchmarks . . . . .	35
7.1.1	Evaluation as a RDF based Reasoner . . . . .	36
7.1.2	Evaluation as a Rule Engine . . . . .	36
8	EVALUATION OF UNIFYING SEARCH AND REASONING FROM THE VIEWPOINT OF GRANULARITY	38
8.1	Quality of Service for Reasoning . . . . .	38
8.2	Evaluation of the Starting Point Strategy . . . . .	38
8.3	Evaluation of the Multilevel Completeness Strategy . . . . .	39
8.4	Evaluation of the Multiperspective Strategy . . . . .	39
8.5	Evaluation Function for Combined Strategies . . . . .	39
9	ANALYSIS OF PARALLEL REASONING AND DISTRIBUTED REASONING	40
9.1	Problem and motivation . . . . .	40
9.2	Conception . . . . .	40
9.3	Technical solution and implementation . . . . .	43
9.4	Distributed execution . . . . .	43
9.5	Parallel execution . . . . .	44
10	TESTING REASONERS WITH HUMAN NORM DATA	45
10.1	Human Gold Standards . . . . .	45
10.2	Crossvalidation with ontologies . . . . .	45
10.3	Crossvalidation with Human norms . . . . .	47
10.3.1	Category norms . . . . .	48
10.3.2	Feature norms . . . . .	48
10.4	Discussion . . . . .	51
11	CONCLUSION	54
	REFERENCES	54



## LIST OF FIGURES

2.1	The LarKC Platform Architecture . . . . .	12
3.1	Workflow of evaluation. . . . .	18
3.2	Workflow of benchmarking . . . . .	19
4.1	Alpha Urban LarKC high level overview. . . . .	22
5.1	Linked Life Data Integration Workflow. . . . .	27
5.2	Linked Life Data mapping rules. . . . .	27
9.1	Some examples of nested parallelism in a reasoner's workflow . . . . .	41
9.2	SCMD workflow . . . . .	42
9.3	MCSD workflow . . . . .	42
9.4	MCMD workflow . . . . .	42





## LIST OF ACRONYMS

<b>Acronym</b>	<b>Description</b>
DL	Description Logics
MORE	Multi-version Ontology Reasoner
OWL	Web Ontology Language
PION	The System of Processing Inconsistent Ontologies
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SPARQL	SPARQL Protocol And RDF Query Language



## 1. Introduction

The essence of the LarKC project is to go beyond notions of absolute correctness and completeness in reasoning. We are looking for retrieval methods that provide useful responses at a feasible cost of information acquisition and processing. Therefore, generic inference methods need to be extended to non-standard approaches. In consequence, traditional metrics such as completeness and correctness for reasoning need to be replaced by metrics that ratio the utility of solutions with their related costs as a means to evaluate the chosen problem solver. Therefore, we will develop a framework for evaluation and measuring the relative utility of various reasoning approaches that will be implemented in this project.

Approximate reasoning is non-standard reasoning which is based on the idea of sacrificing soundness or completeness for a significant speed-up in reasoning. This is to be done in such a way that the loss of correctness is at least outweighed by the obtained speed-up [33]. Anytime reasoning in which more answers can be obtained over time is expected to be a behaviour of approximate reasoning for the LarKC platform. Granular reasoning is considered as some kind of approximate reasoning, in which multiple perspectives/views can be selected for reasoning with variable fine-grained data. Rule-based reasoning provides the facilities for extending ontology reasoning with rules. Parallel reasoning and distributed reasoning are considered to be essential for Web scale reasoning to improve the scalability. All those non-standard reasoning approaches need new metrics and frameworks for the evaluation and benchmarking of reasoners developed within the LarKC platform.

Streaming data poses unique challenges to the design of a benchmark. For queries over this data to be meaningful, the input data must have semantic validity and not just be random. As most stream queries are continuous, performance metrics should be based on response time rather than completion time. The benchmark must be verifiable even though results returned may vary depending on when they are generated. The database community elaborated in 2004 a stream data management benchmark named Linear Road. Linear Road simulates an urban expressway system where tolls are determined according to such dynamic factors as congestion, and accident proximity. In LarKC we intend to extend the Linear Road benchmark in order to stress the reasoning capabilities of LarKC platform when selection, abstraction, and reasoning plug-ins based on stream database techniques are used together. Such extension will also benefit the Urban Computing use case.

These new reasoning paradigms, which fuse approaches from many different fields will also require new approaches to evaluation. Traditional measures like soundness and completeness will have to be enriched with measures such as recall and precision, and worst-case complexity will have to be enriched by approaches such as anytime performance profiles. In this document, we will develop such new evaluation measures and apply them to the implemented reasoning plug-ins, both on synthetic datasets and on datasets from the use-cases.

A selection of the developed reasoning methods will be deployed in the case-studies, as determined by requirement analysis and experiments in the use-cases. An evaluation of the deployed reasoning plug-ins will be provided by the end of project. The extended Linear Road benchmark is split into three subsets that stress the selection, the abstraction and the reasoning aspects of LarKC. The obtained reasoning benchmark will be used in evaluating the reasoning on stream database techniques.

In this document, we will develop the initial framework of evaluation and benchmarking of reasoners deployed with the LarKC platform. We will define the evaluation



methods, measures, benchmarks, and performance targets for the plug-ins to be developed in various tasks such as approximate reasoning with interleaved reasoning and selection and rule-based reasoning. These will be examined with the perspectives of the use-cases of WP6 and WP7, as well as in terms of synthetic benchmarks. Since stream reasoners have not yet been developed at the first year of the LarKC project, we will investigate the evaluation and benchmarks of stream reasoners in subsequent deliverables.

We will explore the evaluation and benchmarking of reason plug-ins in subsequent deliverables such as D4.7.2 entitled “Evolved Evaluation and Revision of plug-ins deployed in use-cases”, which will be released at Month 33, and D4.7.3 entitled "Final Evaluation and Revision of plug-ins deployed in use-cases", which will be released at Month 42.

This document is organized as follows. In Chapter 2, we overview the LarKC Platform and the general picture of reasoner plug-ins, which will be developed or deployed within the LarKC platform. In Chapter 3, we develop a framework of evaluation and benchmarking of reasoners. Chapter 4 and Chapter 5 examine the evaluation issues from the perspectives of the use cases in the LarKC project. Chapter 6 explores the evaluation of approximate reasoning. Chapter 7 describes the initial idea of the evaluation of rule-based reasoners. Chapter 8 explores the evaluation of granular reasoning. Chapter 9 discusses the ideas on the evaluation of parallel reasoning and distributed reasoning. Chapter 10 provides some ideas about the evaluation from the psychological and cognitive perspectives. Chapter 11 concludes the document.

## 2. LarKC Platform

### 2.1 LarKC Architecture

In [45], the first version of the LarKC architecture has been proposed. This design is based on a thorough analysis of the requirements and considering the lessons learned during the first year of the project. Figure 2.1 shows a detailed view of the LarKC Platform architecture.

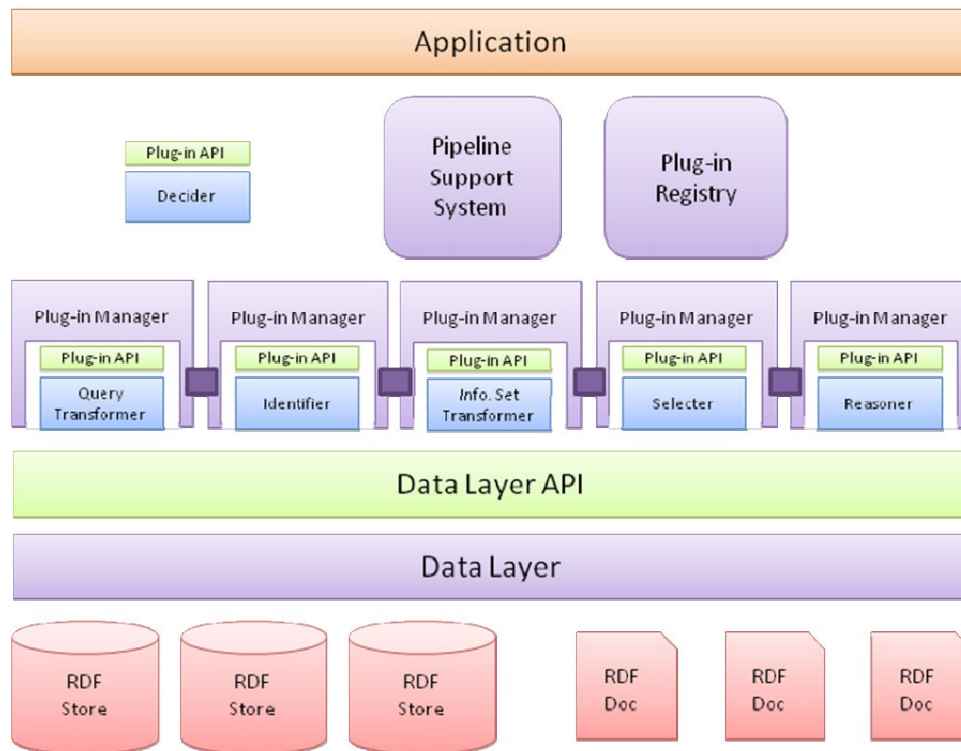


Figure 2.1: The LarKC Platform Architecture

The LarKC platform has been designed in a way so that it is as lightweight as possible, but provides all necessary features to support both users and plug-ins. For this purpose, the following components are distinguished as part of the LarKC platform:

- **Plug-in API:** defines interfaces for plug-ins and therefore provides support for interoperability between platform and plug-ins and between plug-ins.
- **Data Layer API:** provides support for data access and management.
- **Plug-in Registry:** contains all necessary features for plug-in registration and discovery
- **Workflow Support System:** provides support for plug-in instantiation, through the deployment of plug-in managers, and for monitoring and controlling plug-in execution at workflow level.
- **Plug-in Managers:** provides support for monitoring and controlling plug-ins execution, at plugin level. An independent instance of a Plug-in Manager is



deployed for each plug-in to be executed. This component includes the support for both local and remote execution and management of plug-ins.

- **Queues:** provides support for deployment and management of the communication between platform and plug-ins and between plug-ins.

## 2.2 LarkKC reasoning plug-ins

### 2.2.1 Reasoning APIs

All LarKC plug-ins share a common super class, namely the Plugin class. This class provides an interface for functions common to all plug-in types. The functions provided by the Plugin interface can be seen in Table 2.1.

Table 2.1: Plugin Interface

Function name	Return type
getIdentifier()	String
getMetaData()	MetaData
getQoSInformation()	QoSInformation
setInputQuery(Query theQuery)	void

All plug-ins are identified by a name, which is a string. Plug-ins provide meta data that describes the functionality that they offer. Plug-ins provide Quality of Service (QoS) information regarding how they perform the functionality that they offer. All plug-ins may need access to the initial query (entry query in the LarKC platform) and thus a mutator is provided by specifying this query.

The reasoning plug-in will execute a given SPARQL Query against a Triple Set provided by a Select plug-in. The interface of the reasoning plug-in can be seen in Table 2.2.

Table 2.2: Reasoner Plug-in Interface

Function name
sparqlSelect(SPARQLQuery q, SetOfStatements s, Contract c, Context ctx)
sparqlConstruct(SPARQLQuery q, SetOfStatements s, Contract c, Context ctx)
sparqlDescribe(SPARQLQuery q, SetOfStatements s, Contract c, Context ctx)
sparqlAsk(SPARQLQuery q, SetOfStatements s, Contract c, Context ctx)

The reasoning plug-in supports the four standard methods for a SPARQL endpoint, namely select, describe, construct, and ask. The input to each of the reason methods are the same and includes the query to be executed, the statement set to reason over, the contract, which defines the behavior of the reasoner, and the context, which defines the special information of the reasoning task. The output of these reasoning methods depends on the reasoning task being performed. The select method returns a Variable Binding as output where the variables correspond to those specified in the query. The construct and describe methods return RDF graphs, in the first case this graph is constructed according to the query and in the second the graph contains triples that describe the variable specified in the query. Finally ask returns a Boolean Information Set as output, which is true if the pattern in the query can be found in the Triple Set, or false if not.



### 2.2.2 Reasoning Plug-ins

The LarKC reasoning plug-ins can range from the reasoners which provide the standard reasoning support with RDF/RDFS/OWL data to the reasoners which realize non-standard reasoning tasks such as reasoning with inconsistent ontologies, rule-based reasoning, stream reasoning. Here is an (incomplete) list of the LarKC reasoning plug-ins which have been developed or may be developed for the LarKC platform.

- **SPARQL Query Evaluation Reasoner:** This reasoning plug-in wraps OWL-IM and enables the execution of SPARQL Select, Construct, Describe and Ask queries to be executed against it.
- **Pellet Reasoner:** This reasoning plug-in is a wrapper of Pellet SPARQL DL Reasoner<sup>1</sup>, which provides the reason support of Description Logics.
- **DIG Interface:** This reasoning plug-in provides the support for the DIG interface<sup>2</sup>, which allows an external DIG reasoner to be called, like RACER, FACT++,KAON2, PION, etc.
- **OWLLink Interface:** This reasoning plug-in provides the support for the OWLLink interface, a new generation of the DIG interface, which is designed to support reasoning with OWL data.
- **OWLAPI Reasoner:** This reasoning plug-in provides the support for OWL APIs, which is a standard for reasoners with OWL data.
- **PION Reasoner:** This is a reasoner which can be used for reasoning with inconsistent ontologies. Namely, given an inconsistent ontology and a query, the PION reasoner can return a meaningful answer.
- **IRIS Rule Reasoner:** This is a a rule-based reasoner configurable with IRIS inference rules.
- **Stream Reasoner:** This reasoning plug-in provides the support for stream reasoning.
- **OpRes Path Finder Reasoner:** A reasoner to apply graph algorithms (in this case Dijkstra) to find the "desirable path" on graphs modeled with RDF (with a fixed schema).
- **OB Path Finder Reasoner:** A reasoner to compute the shortest path with OntoBroker on graphs modeled in RDF.
- **Cyc Reasoner :** A reasoning plug-in, which transfers reasoning requests to the ResearchCyc reasoner. Currently it has limited SPARQL to CycL / RDF to CycKb support.

---

<sup>1</sup><http://clarkparsia.com/pellet>

<sup>2</sup><http://dig.sourceforge.net/>



## 3. A Framework of Evaluation and Benchmarking for Ontology Reasoners

### 3.1 General Consideration

In this chapter, we will develop a framework of evaluation and benchmarking for ontology reasoners. The main idea is to use the framework, which have developed in the KnowledgeWeb project for benchmarking inconsistency reasoners in the Semantic Web[22]. In LarKC, we can use this framework to evaluate and benchmark both standard reasoner plug-ins and non-standard reasoner plug-ins, e.g. the PION reasoner for reasoning with inconsistent ontologies, and the MORE reasoner plug-in for reasoning with multi-version ontologies.

In ontology engineering, evaluation and benchmarking target software products, tools, services, and processes. The objects are called tested systems. Evaluation and benchmarking are the systematic determination of merit, worth, and significance of tested systems. Those merit, worth, and significance are characterized as a value relation, which is considered as a preference relation, i.e., a partial order set  $\langle A, \succ \rangle$ . We consider a tested system as one which is targeted by the objectives of evaluation or benchmarking. A tested system can be characterized as an input-output function, alternatively, called a characteristic function of the tested system. Namely, it maps a tuple of the input parameters into an output value. A value relation is defined as a preference relation on a set of values. Namely, a value relation is characterized as a partial order set.

We define *evaluation* as the systematic determination of the values of tested systems with respect to its partial ordered value relation, whereas *benchmarking* as a continuous process for improving by systematically evaluating tested systems, and comparing them to those considered to be the best. Namely, benchmarking is the continuous process of evaluation.

In the LarKC project, Task 4.7.1 is requested to develop an initial framework for evaluation of reasoner plug-ins. For the purpose of continuous improvement of the LarKC platform, the issue of benchmarking reasoner plug-ins should be also covered.

### 3.2 Goals and Criteria for Evaluation and Benchmarking of Reasoner Plug-ins

We consider the following initial goals for evaluating LarKC Reasoner plug-ins:

- **Bug Detection:** A good evaluation of reasoner plug-ins should be able to detect hidden bugs in the implementation. These bugs may be hard to detect with manual examination by developers. It requires that test data sets cover many functionalities/use cases of reasoner plug-ins.
- **Robustness:** A robust reasoner plug-in should not fail with noisy or erroneous test data. Thus, special test data sets should be designed to test the robustness of a reasoner plug-in.
- **Performance analysis:** One of the main concerns on the quality of a reasoner plug-in is its performance. Thus, a necessary procedure of evaluation and benchmarking of reasoner plug-ins is to provide an analysis of their performance.



The usual criteria for examining the performance of reasoner plug-ins are: (i) the time costs, including the time cost for getting the first query answer with anytime behavior, and the average time cost for each query answer, (ii) the resource consumption, including the maximal working memory request, and (iii) the quality of the query answers, which will be discussed in the next section.

- **Scalability Potential:** The LarKC platform is expected to support Web scale reasoning. Thus, the scalability of a reasoner plug-in becomes a crucial issue for the performance of the overall platform. The scalability potential of a reasoner is how well it can deal with large amount of data.
- **Platform Improvement:** A useful evaluation and benchmarking of reasoner plug-ins should be able to find bottle necks within the platform. It would provide an analysis of how the design of platform can be improved.

### 3.3 Measuring the Quality of Query Answers

As discussed in the last section, the quality of query answers is one of the main criteria for evaluating and benchmarking of reasoners.

The answer value set for standard ontology reasoning is usually considered as a Boolean value set, namely, it consists of ‘true’ and ‘false’. The answer value set for reasoning with inconsistent ontologies usually consists of three values *accepted*, *rejected*, and *undetermined*, as introduced in the PION system. We will develop gold standards, which represents intuitive answers from a human for queries on reasoning with consistent or inconsistent ontologies. Thus, we can compare the answers from the tested system/approach with the gold standard, which is supposed to be intuitively true by a human to see to what quality of query answers provided by tested systems.

For a query with an inconsistent ontology, there might exist the following difference between an answer from the tested system/approach and its intuitive answer in a gold standard.

- **Intended Answer:** the system’s answer is the same as the intuitive answer;
- **Counter-intuitive Answer:** the system’s answer is opposite to the intuitive answer. Namely, the intuitive answer is ‘accepted’ whereas the system’s answer is ‘rejected’, or vice versa.
- **Cautious Answer:** The intuitive answer is ‘accepted’ or ‘rejected’, but the system’s answer is ‘undetermined’.
- **Reckless Answer:** The system’s answer is ‘accepted’ or ‘rejected’ whereas the intuitive answer is ‘undetermined’. We call it a reckless answer, because under this situation the system returns just one of the possible answers without seeking other possibly opposite answers, which may lead to ‘undetermined’.

Therefore, a value set

$\{intended\_answer, cautious\_answer, reckless\_answer, counter\_intuitive\_answer\}$ ,

can be introduced for the evaluation of answers with golden standards. An intended answer is considered as a best one, whereas a counter intuitive answer is considered as a worse one. Cautious answers are usually not considered as wrong answers, whereas





reckless answers may give wrong answers. Thus, a preference relation on the value set can be like this:

$$\{intended\_answer \succ cautious\_answer, \\ cautious\_answer \succ reckless\_answer, \\ reckless\_answer \succ counter\_intuitive\_answer\}$$

Based on this preference order, we can measure the quality of query answers by the following answer rates:

- **IA Rate**, which counts only intended answers. Namely the Intended Answer Rate is defined as the ratio of the amount of Intended Answers to the total amount of the answers.
- **IC Rate**, which counts non-error answers. Namely, IC Rate = (Intended Answers + Cautious Answers) / TotalAnswerNumber.

### 3.4 Workflows of Evaluation and Benchmarking

Common data sets and common golden standards are usually used for an evaluation of different tested systems. Those systems may be heterogeneous with respect to their input data. For example, a reasoner may support only OWL data, whereas another reasoner may support only DIG data. Therefore a data translator is needed to convert data sets represented in a standard format into the data sets which are represented in a format that is supported by a tested system. Based on a comparison between test results and golden standards, result evaluation can be done manually, semi-automatically, or automatically. The output of the result evaluation and the implication are further analyzed by an evaluation analysis. The methods of statistics and visualization are usually introduced in this phase for better illustration. The evaluation results will be ranked with respect to its value relation. Finally, it leads to an evaluation report which concludes the values of tested systems and explain the reasons why the system behaves differently. An investigation is usually made to detect the problem of tested systems based on the analysis of the evaluation. The workflow of evaluation is shown in Figure 3.1.

As discussed above, benchmarking is a continuous processing of evaluation. Therefore for benchmarking, evaluation results are used further for the improvement of tested systems. This would usually lead to new versions of tested systems. Based on a benchmarking analysis, new test data sets may be re-designed or previous data sets are adjusted for further evaluation with respect to some targeted problems. The workflow of benchmarking is shown in Figure 3.2.

### 3.5 A Specification Language for Golden Standards

Manual evaluation and analysis of test results are usually time consuming, labor intensive, and error prone. The formalism of golden standard will pave a way for automatic or semi-automatic evaluation and analysis of test results.

A golden standard is an evaluation function which maps queries into answers with confidence values. For reasoner benchmarking, a gold standard is a (partial) function which maps queries into (intuitive) answers with a confidence value. For example, for benchmarking inconsistency processing, we considered the answer set

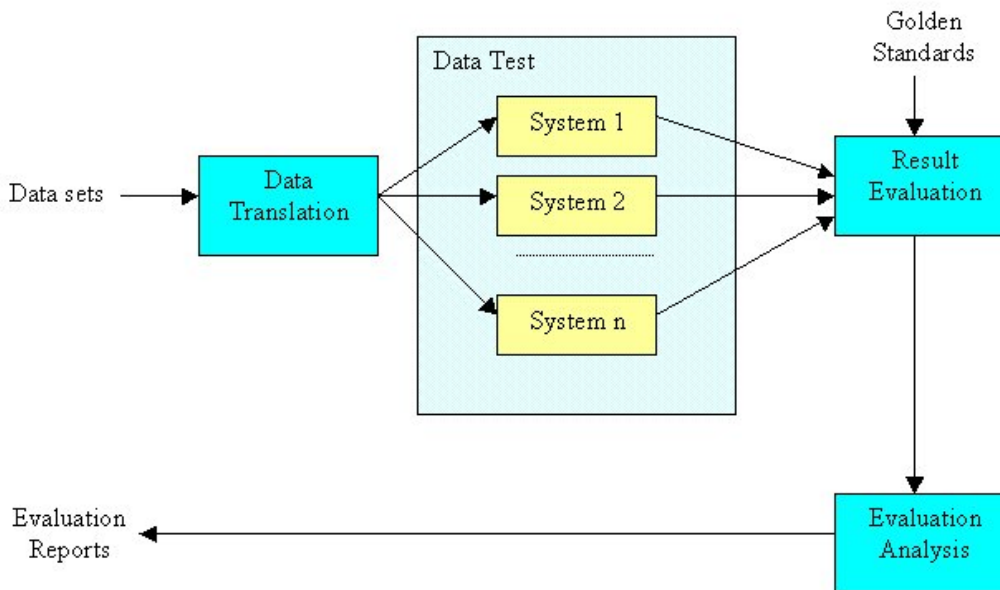


Figure 3.1: Workflow of evaluation.

$\{accepted, rejected, undetermined\}$ . For a query "are birds animals?", the expected answer is intuitively considered as "accepted" with confidence value "1.0". However, for the query "are men animals?", the expected answers may be well suitable to be specified as an answer with lower confidence value, say, "accepted" with confidence value "0.4", "rejected" with confidence value "0.4", and "undetermined" with the confidence value "0.2". Namely, we use the confidence values to represent some kinds of uncertainty of expected answers. The confidence values can be obtained by various approaches, like from questionnaires, statistics, machine learning, etc.

We design gold standards which are independent from a specific ontology. Namely, it is up to evaluators/users to decide which ontologies can be applied with respect to a golden standard.

In the following, we develop a golden standard specification language which is suitable for SPARQL queries as reasoning queries. Thus, it is an XML file, which is easy to use and read. Table 3.1 shows an example of a golden standard which is encoded as an XML document.

This XML document specifies the name and the version of the golden standard. Each query consists of a detailed query statement (in the SPARQL query language) and its expected answer specification. Each expected answer is attached by a confidence value. For non-Boolean answers, like those for sparqlSelect and sparqlConstruct queries which would return a variable binding or a rdf graph as an answer, the values of expected answers are specified with a detailed xml-encoded subtree as specified in Table 3.2

Alternatively, the golden standard specification language can be defined by using the standard meta data languages or ontology language, usch as RDF/RDFS/OWL. Namely, a gold standard can be specified as meta data or an ontology, which provides a possibility for reasoning with gold standards. Table 3.3 shows an example of the RDF representation of the gold standards.

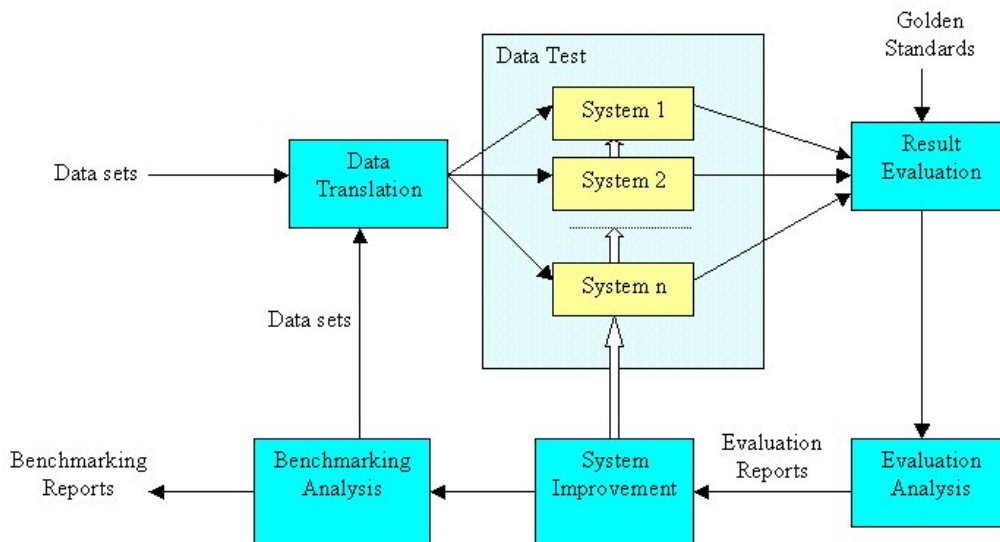


Figure 3.2: Workflow of benchmarking

Table 3.1: Example of Golden Standard

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<goldenStandard xmlns="http://wasp.cs.vu.nl/larkc/d471/lang"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wasp.cs.vu.nl/larkc/d471/gd.xsd">
<name value="LarKC golden standard example 1" version="0.0.1"/>
<comment text="just an example, which is independent from any
  particular ontology. It is up to evaluators to decide
  which ontology can be applied"/>
  <query id="Are birds animals?" querytype="subsumes">
    <queryBody> ... </queryBody>
    <expectedAnswers>
      <answer value="accepted" confidence="1"/>
    </expectedAnswers>
  </query>
  <query id="Are men animals?" querytype="subsumes">
    <queryBody> ... </queryBody>
    <expectedAnswers>
      <answer value="accepted" confidence="0.4"/>
      <answer value="undetermined" confidence="0.2"/>
      <answer value="rejected" confidence="0.4"/>
      <comment text="just an example which shows the possibility
        of multiple answers in a golden standard"/>
    </expectedAnswers>
  </query>
</goldenStandard>
```



---

Table 3.2: XML Format for Expected Answers

---

```
...
<query id="List all the subconcepts fo wine"
      querytype="sparqlSelect">
  <queryBody>
    <sparqlPrefix name="rdfs"
      value="http://www.w3.org/2000/01/rdfschema#/">
    <sparqlPrefix name="wine"
      value="http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#/">
    <sparqlBody value="SELECT ?X WHERE {?X rdfs:subClassOf wine:Wine.}"/>
  </queryBody>
  <expectedAnswers>
    <answer confidence="1.0"/>
    <value>.....</value>
  </expectedAnswers>
...

```

---



Table 3.3: RDF representation of the gold standards

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:fields="http://sindice.com/vocab/fields#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:larkc="http://www.larkc.eu/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdf:Description
    rdf:about="http://wasp.cs.vu.nl/larkc/d471/lang#GoldStandardName">
  <larkc:name>LarkC golden standard example 1</larkc:name>
  </rdf:Description>
  <rdf:Description
    rdf:about="http://wasp.cs.vu.nl/larkc/d471/lang#Query">
    <larkc:queryID>Are birds animals?</larkc:queryID>
    <larkc:queryType>subsumes</larkc:queryType>
    <larkc:queryBody>...</larkc:queryBody>
    <larkc:expectedAnswers>
      <rdf:Bag><rdf:li><larkc:answer>
        <larkc:value>accepted</larkc:value>
        <larkc:confidence>1</larkc:confidence></larkc:answer></rdf:li>
      </rdf:Bag>
    </larkc:expectedAnswers>
  </rdf:Description>
  <rdf:Description
    rdf:about="http://wasp.cs.vu.nl/larkc/d471/lang#Query">
    <larkc:queryID>Are men animals?</larkc:queryID>
    <larkc:queryType>subsumes</larkc:queryType>
    <larkc:queryBody>...</larkc:queryBody>
    <larkc:expectedAnswers>
      <rdf:Bag><rdf:li><larkc:answer>
        <larkc:value>accepted</larkc:value>
        <larkc:confidence>0.4</larkc:confidence></larkc:answer>
      </rdf:li>
      <rdf:li><larkc:answer>
        <larkc:value>undetermined</larkc:value>
        <larkc:confidence>0.2</larkc:confidence></larkc:answer>
      </rdf:li>
      <rdf:li><larkc:answer>
        <larkc:value>rejected</larkc:value>
        <larkc:confidence>0.4</larkc:confidence></larkc:answer>
      </rdf:li></rdf:Bag>
    </larkc:expectedAnswers>
  </rdf:Description>
  .....
</rdf:RDF>
```

## 4. Evaluation from a WP6 Perspective

### 4.1 Alpha Urban LarKC

The “Alpha Urban LarKC” (AUL) is the first implementation, based on the first release of the LarKC platform, of an end-to-end application for the Urban Computing use case (WP6).

In Figure 4.1 we provide an high level overview of AUL. Users of AUL are located in a (potentially unknown) city and would like to organize a day of visiting some places, meeting friends, attending a musical concert, etc. They needs to find interesting destinations (such as monuments or relevant places in the city or events that take places in the city) and to understand the most suitable way to reach them.

To solve the problem today, the user would have to use multiple applications, and manually pass intermediate results from one service to another. The current implementation of AUL searches for monuments of Milan published in DBpedia<sup>1</sup> using the Sindice<sup>2</sup> API<sup>3</sup>; it searches for events in Milan using the Eventful<sup>4</sup> API<sup>5</sup>; and it compute the most suitable way to reach them using the topology of the streets of Milan provided by the Milan Municipality<sup>6</sup>. To do so, AUL uses the LarKC platform configured with three alternative workflows with one common decider. Two of such workflows select destinations (either monuments or events), the third finds the paths to such destinations. The decider chooses which workflow to run depending on the query issued by the the AUL client.

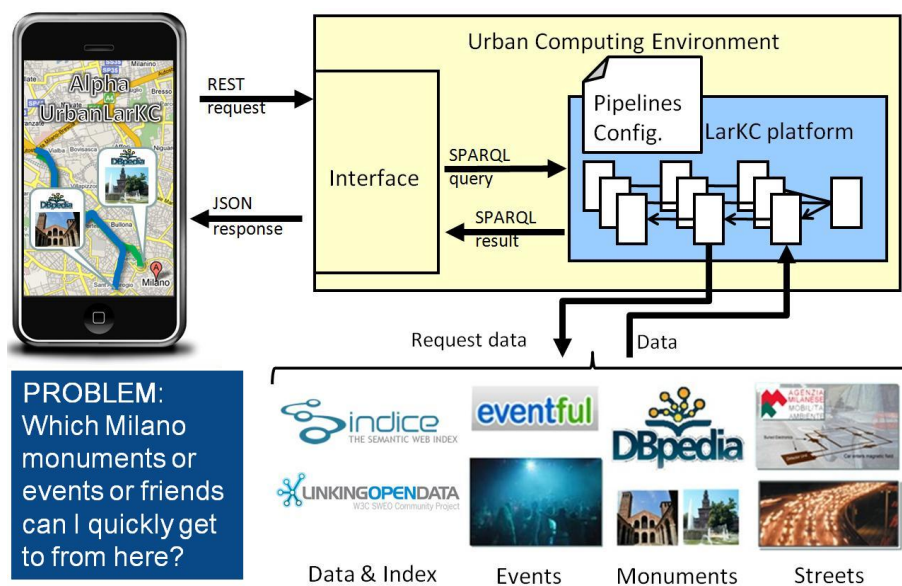


Figure 4.1: Alpha Urban LarKC high level overview.

<sup>1</sup><http://dbpedia.org/About>

<sup>2</sup><http://sindice.com/>

<sup>3</sup><http://sindice.com/developers/api>

<sup>4</sup><http://www.eventful.com/>

<sup>5</sup><http://api.eventful.com/>

<sup>6</sup><http://www.ama-mi.it/documenti/default.asp>



## 4.2 Workflows of Alpha Urban LarKC

The two workflows selecting destinations are variants of the “SPARQL for the Web” workflow<sup>7</sup>). They use dedicated search engines (Sindice in the case of monuments and Eventful in the case of events) to identify relevant information on the Web, they transform it into RDF if it is not yet in such a form, they put it in the LarKC data layer, and they use the SimpleSparqlReasoner to answer the query issued by the client. The quality and quantity of results is therefore largely determined by the precision and recall of the two search engines used to identify relevant information.

The evaluation of those two workflows (see also D6.6 “2nd periodic report on data and performances” [10]) is twofold.

1. Functional evaluation, meaning the ability to provide useful answers to the users in reasonably short time. For this reason we intend to measure:
  - *maximum, minimum, average* response time. Notably, the response time of the workflows depends not only on LarKC, but also on the latency of the network and the possible variable response time of the external services (i.e., sindice and eventful).
  - *maximum, minimum, average* percentage of “dropped” results, i.e., potential results to the query that were discarded by the workflow. Reasons for the workflow to “drop” results are numerous. So far we observed three case: external service was not available, retrieved information is not sufficient to answer the query, and retrieved information would be sufficient if it was coded in a different way. Other reasons are approximation and loss of completeness or soundness.
  - *maximum, minimum, average* percentage of wrong results. For instance, the user asks for Milan (Italy) monuments and gets Milan (Texas) monuments or retrieved location of events is not the real one of the venue. Appropriate reasoning task (not in place in the current version) may reduce such percentage.
  - *maximum, minimum, average* percentage of duplicated results. For instance, the same real world event appears several time in the result set with slightly different description.
2. Stress test evaluation, meaning the ability to keep an adequate level of service when an increasing number of concurrent users performs requests. For this reason we intend to measure how the functional indicators depend on the number of concurrent users.

The third workflow, which finds the paths to the destinations, is more complex and comes in three different varieties (see also D6.3 [8]). It uses several different strategies to identify the minimum subset of Milan streets needed to find the requested path, it loads such subsets into the LarKC data layer and invokes a “path finding service”, which is wrapped as a Reasoner plug-in. For this reason we believe that the most meaningful evaluation is measuring the computational overhead introduced by the LarKC platform. Our evaluation strategy is to compare LarKC performances in finding the most suitable way to go from one place to another with a “term of comparison”, a

---

<sup>7</sup>see <http://wiki.larkc.eu/LarkcPlugins#Pipelineconfigurationsthatusetheplugins> for more details.



software product which achieves the same solution without using LarKC technologies. More details are available in Section 2.3 of deliverable D6.3 [8].

## 4.3 Use Cases for Reasoning Plug-ins

The performance of the workflows heavily depend on the performance of reasoner plug-ins. Thus, it is very useful to obtain the performance information of reasoner plug-ins by the evaluation of those plug-ins. Before we move to discuss how various reasoning plug-in can be tested and evaluated, we would like to discuss various use cases of reasoner plug-ins which may be deployed in the urban computing case study.

### 4.3.1 Standard RDFS/DL Reasoners

Standard RDFS/DL/OWL reasoners are used to obtain implicit knowledge which is hidden inside the data. For example, in the scenarios of urban computing, one may query or search for monuments or castles which are located around central Milan. A standard OWL reasoner like Pellet can be used to reason with Geoname data<sup>8</sup> to check whether or not a location (e.g., Castello Sforzesco) is located at the central Milan or outside the center, based on its latitude (N 45° 28' 11") and longitude (E 9° 10' 48") with the definition of central Milan.

### 4.3.2 Approximation Reasoners

Approximation reasoners are used to obtain approximate answers, which are not necessarily always sound. Take the same example above about searching for monuments or castles which are located around central Milan. An approximate reasoner can provide some estimated probably correct answers, like Castello Sforzesco, just based on its latitude and longitude, when it lacks a clear information about the latitude and longitude of central Milan.

### 4.3.3 Rule-based Reasoners

The rule based reasoner plugin, based on the IRIS datalog reasoner<sup>9</sup>, can be used for two purposes within Alpha Urban LarKC. The first potential application is to perform (possibly a custom subset of) standard RDFS/OWL inference. Equipped with a specific set of inference rules the reasoner plugin can thus be an alternative to standard OWL reasoners such as Pellet. The second potential use of the rule based reasoner plugin (being actually a Datalog engine at its heart), is for the computation of reachability between two locations. However, closer investigation is required to determine the actual performance of the reasoner plugin when compared to a highly specialized implementation.

### 4.3.4 Parallel and Distributed Reasoners

When thinking about reasoning over large amount of data, the scalability and performance characteristics are important for the workflow execution. Distributed and

---

<sup>8</sup><http://www.geonames.org>

<sup>9</sup><http://www.iris-reasoner.org/>





parallel reasoning are well-recognized approaches for improving performance and scalability of the reasoning process, due to enabling diverse high-performance resources for the plug-in execution. The optimal resource load-balancing is the most important factor for reaching the maximal performance and scalability effect due to the distribution and parallelization. For this purpose, special techniques for avoiding idle times, bottlenecks and redundancy should be considered.



## 5. Evaluation from a WP7a Perspective

### 5.1 Linked Life Data

The work packages WP7a "Semantic Integration for Early Clinical Development" and WP7b "Carcinogenesis Reference Production" require a massive data integration effort to bring closer disconnected and semantically related data sources. Linked Life Data is a RDF warehousing solution developed in the context of WP7a and also used in the WP7b use case [2], [32]. It integrates distributed information sources that increase the context of the knowledge and support the execution of complex data analysis queries. Based on the experience of integrating more than 20 different datasets, several pragmatic approaches are undertaken in order to achieve the final goal:

- **Align instances that are equivalent:** No single authority or best practice policy or convention define how to identify biomedical entities. Thus, many of the data sources distributed in RDF format use different namespaces or patterns to generate the URI from the original database identifiers.
- **Connect instances with specific forms of semantic relationships:** The network of biological databases specifies a number of semantic relationships. For instance, the meta-thesauri of UMLS <sup>1</sup> and NCI <sup>2</sup> are described by a strict semantic network. Another notable example are the OBO format ontologies <sup>3</sup> that could be mapped to SKOS [27].
- **Generate semantic annotations that map textual data with the RDF model:** Data sources like PubMed <sup>4</sup> or Linked CT <sup>5</sup> contain large sections of unstructured textual information. The application of a Named Entity Recognition (NER) process plays an important part in linking structured with unstructured information and finding correlations between the data.

### 5.2 Semantic Data Integration Workflow

The process of loading and generation of new knowledge in Linked Life Data is a multiphase workflow. It shares many similarities with other data warehousing solutions that require complex Extraction, Transformation, and Loading (ETL) scripting. First, the knowledge base is created from different data source snapshots. Then it is transformed to a common data model (RDF). An instance level alignment is applied to identify the redundant entities. New implicit knowledge is then derived with a lightweight inference schema. Finally, a massive parallel information extraction process is performed to enrich the semantic network with statements generated as a result of the text-mining.

An important feature of the Linked Life Data approach is the execution of ETL scripts against a formal semantic model. We have defined 6 integration rules to link different ontology instances:

---

<sup>1</sup><http://www.nlm.nih.gov/research/umls>

<sup>2</sup><http://ncit.nci.nih.gov>

<sup>3</sup><http://www.obofoundry.org>

<sup>4</sup><http://www.ncbi.nlm.nih.gov/PubMed>

<sup>5</sup><http://linkedct.org/index.html>

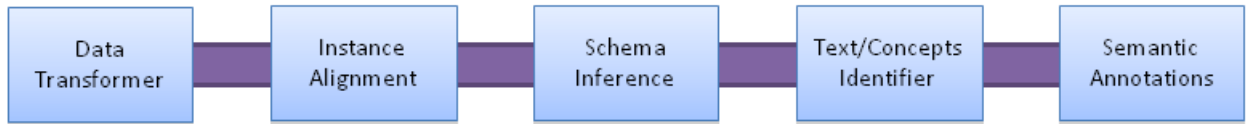


Figure 5.1: Linked Life Data Integration Workflow.

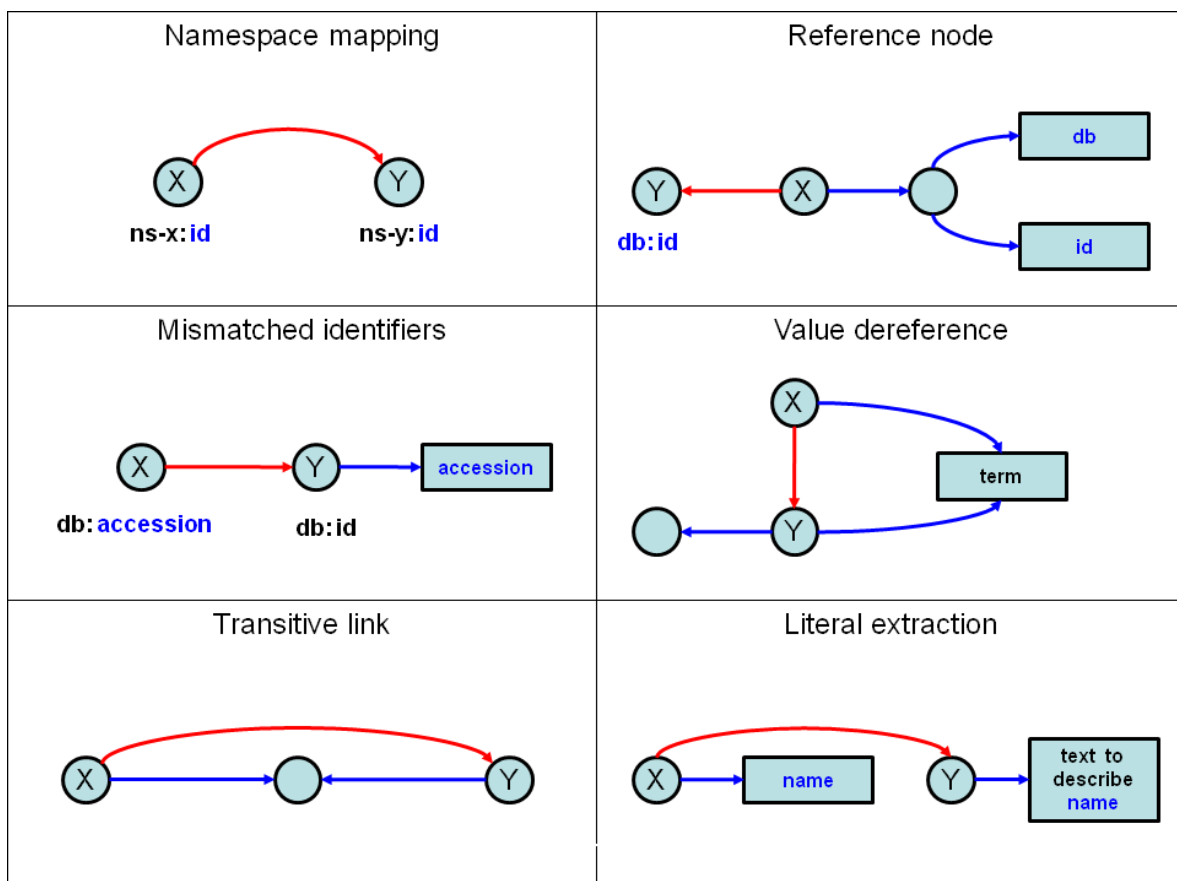


Figure 5.2: Linked Life Data mapping rules.



1. **Namespace mapping:** Two RDF datasets use one and the same local identifier but define different namespaces; This case is typical for data sources in RDF format that refer to common database identifiers like GO, Entrez-Gene, etc., which have no resolvable URI supported by the authors.
2. **Reference node:** A trick to prevent the former pattern that uses a reference dummy node to designate the database id and name; This is also the recommended way to create cross-references to external data sources in the BioPAX specification. However, the nodes remain disconnected even after the data sources were imported.
3. **Mismatched identifier:** Database entries have multiple identifiers used for different purposes; For example, the EntrezGene database has a gene symbol (alphanumeric string) and an id (numeric value). The id could be regarded also as a composite key constituted by a unique combination of gene symbol and organism.
4. **Value dereference:** A lazy way to reference controlled vocabularies by using only the concept name, and not the identifier; PubMed is annotated with the MeSH term names, but not the MeSH concept ids.
5. **Transitive link:** A pattern used to link two data sources based on the common relation to a third-one; DBpedia has links to Freebase and ICD-10 codes.
6. **Literal extraction:** A pattern used to link resources based on literals that enumerates a list of named entities; DrugBank indication field lists a sequence of diseases.

### 5.3 Evaluation criteria

Linked Life Data data integration workflow could be a very computationally and input/output insensitive process, if executed against big datasets. In the first prototype version of Linked Life Data, the input datasets were more than 2 billion RDF explicit statements. In its final version, the prototype must be able to handle regular incremental updates of data sources that could be as frequent as every day for PubMed.

Data source	RDF statements	Ontology language
Uniprot	1,146,084,021	Protein sequences and annotations
Entrez-Gene	107,193,308	Genes and annotation
Gene Ontology	9,656,074	Gene and gene product annotations
BioGRID	1,892,897	Protein interactions extracted from the literature
NCI	333,415	Human pathway interaction database
The Cancer Cell Map	173,914	Cancer pathways database
Reactome	2,538,793	Human pathways and interactions
INOH	432,456	Pathway database
KEGG	18,128,735	Molecular Interaction
PubMed *	900,861,385	Biomedical citations
UMLS *	79,88,309	Biomedical meta-thesaurus

The WP7a use case will evaluate the existing reasoners with respect to their functional and non-functional parameters:



- Functional properties
  - Reduced dependencies between the different workflow steps - a workflow property that measures the administrative costs to maintain the system, introduces new data sources, and follows the data source evolution.
  - Customization of the system - a property that indicates the cost to maintain different version of Linked Life Data integration workflow
- Non-functional properties
  - Scalability - a parameter that indicates the maximum size of the datasets that could be processed by the Linked Life Data integration workflow.
  - Performance - a parameter that measures the time for completing the Linked Life Data integration workflow; ultimately the workflow should be executed incrementally on a daily basis.

## 5.4 Use Case of Reasoning Plug-ins

### 5.4.1 Description Logic Reasoners

Many of the distributed data sources conform with the standard ontology languages. For instance, the BioPAX 2.0 schema uses OWL-DL or the OBO format could be transformed to SKOS schema. A standard OWL reasoner could be used to infer that "Bronchial Asthma" is a "Respiratory Disease". Another example is that "Chronic bronchitis" is a close match of "Chronic Obstructive Pulmonary Disease".

### 5.4.2 Rule-based Reasoners

Rule-based reasoners are a family of systems that efficiently implement logic programming like:

```
<A> skos:broader <B> .  
<B> skos:broader <C> .
```

entails

```
<A> skos:broaderTransitive <B> .  
<B> skos:broaderTransitive <C> .  
<A> skos:broaderTransitive <C> .
```

In the context of WP7 work package, the rule based reasoning is an efficient way of implementing mapping rules that assists the ontology instance alignment. Another promising case is the implementation of consistency checking rules that validate the model soundness. The data integration process is characterized by a constant flow of new information in the knowledge base (e.g. new versions, additional data sources, etc.). Thus, a special form of controlling the consistency or the correctness of the knowledge base is required. As a minimum, we would need to enforce the consistency rules required by the SKOS specification like:



```
<COPD_Disease> skos:prefLabel "COPD"@en .  
<COPD_Disease> skos:prefLabel "Common Obstructive Pulmonary Disease"@en . - inconsis
```

Another example is the BioPAX domain ontology that uses OWL-DL semantics. From a computational view point, OWL-DL seems unnecessarily complex for multi-billion datasets. In this particular case, the disjoint constructs could be substituted with simple consistency checking rules.

#### 5.4.3 Approximation Reasoners

The approximation reasoners have limited applicability to the WP7 use cases. The scientists would need complete and sound query results, so the approximation reasoners could be used for a quick hypothesis testing.

#### 5.4.4 Parallel and Distributed Reasoners

The WP7 use cases operate with multi-billion datasets. The parallel and distributed processing may decrease the time of loading new data, increase the overall scalability, and support more datasets and richer annotations.



## 6. Evaluation of Approximation Reasoners

In this chapter, we present some general discussion on the foundations for evaluating approximate and anytime reasoning algorithms. We will do this in a very abstract manner, which can be made concrete in different ways, depending on the considered use case. In LarKC deliverable 1.4.1, we will have deeper discussions considering this topic from various aspects.

### 6.1 Introduction and Motivation

The requirements for reasoning services may be quite distinct in various application areas of Web-scale problem solving. In certain fields (as in safety-critical technical descriptions) soundness and completeness are to be rated as crucial constraints, while in other fields less precise answers could be acceptable if this would result in a faster response behavior, since in these kinds of scenarios, users prefer to get a “good enough result” under very limited time. When the data goes to Web scale, this is a very common request.

Although optimizations for very accurate (i.e., sound and complete) reasoning has been developed in many current standard reasoning tools, they do not comply with this kind of approach: in an all-or-nothing manner, they provide the whole answer to the problem after the complete computation. As discussed above, it would be desirable, however, to have reasoning systems which can generate good approximate answers in less time, or even provide “anytime behavior”, which means the capability of yielding approximate answers to reasoning queries during ongoing computation: as time proceeds, the answer will be continuously refined to a more and more accurate state until finally the precise result is reached. Clearly, one has to define this kind of behavior (and especially the notion of the intermediate inaccuracy) more formally.

In the discussion above, we discussed one of the factors that can be considered for approximation algorithms that the user might want to consider to economize on in exchange for a reduced quality of answers. There are many other resources that can be considered to be reduced, here we list some possible directions:

- The amount of memory used by an algorithm (e.g. on memory-limited mobile devices).
- The amount of user interactions needed to complete the task (in order to put less burden on the user).
- The amount of data access needed by the algorithm (e.g. in pay-for-access environments).

From our point of view, the formal framework presented in our research is general enough to cover approximation as a trade-off quality against any resource, time being just a particular (and frequently occurring) example.

The field of knowledge representation has a long history of studying approximate reasoning methods for propositional and first-order logic (see e.g. [12, 20, 36, 34, 9, 42, 16]). These are only now being applied in the context of OWL reasoning for Semantic Web technologies. Notable recent papers in this area are [39, 19, 17, 31, 38] and to the best of our knowledge, this list is almost exhaustive. The methods in these papers are very diverse, and no overall framework exists for formally describing and comparing the quality of these approaches to approximate reasoning.



The notion of approximate reasoning bears two different meanings in two different communities. Often, the notion is associated with uncertainty reasoning, e.g. in the sense of fuzzy or probabilistic approaches. The notion of approximate reasoning we use in this document refers to approximate reasoning algorithms on data that is not uncertain in this sense.

## 6.2 Multiple Forms to Approximate Answers

There are multiple forms that can be considered for approximations. Here we give some illustrative examples and motivations.

- **Approximating the set of answers from the viewpoint of incompleteness:** if not all answers are needed to complete a task (e.g. instead of returning all the phone numbers of a taxi company return only one them, i.e. return a single answer instead of multiple answers to a query).

Or, weaker, if a partial set of answers already allows partial completion of the task. For example consider the scenario where an user is interested in comparing the prices of a given item across various internet shops. The more shops the better, but even a partial set of shops allows you to make a relatively good comparison and decision.

- **Approximating the set of answers from the viewpoint of unsoundness:** when wanting to rapidly exclude a suspected answer: cheaply compute an unsound answer set (= too large); if the suspected answer is not in the computed (too large) answer set, it is certainly not an answer, and can be excluded. This happens when wanting to ensure that no solutions are missed: cheaply compute an unsound answer set (= too large); all correct answers are guaranteed to be included (at the price of having a few extra incorrect answers).

For example, in systems where a rapid diagnosis for high risk faults is required a valid option is to compute an initial unsound (too large) diagnosis. If this set contains a high risk fault then it is worth spending more resources to see if the same high risk fault is included in a more precise diagnosis. As another example consider a query that ask for the current location in the context of location-aware services. This query is an example of the type of queries users may formulate in the Urban Computing use case. A possible solution would be to compute and return an approximate location rather than the exact location. In this case, an unsound but complete answer set would include multiple locations, one of which is the correct one.

- **Approximating individual answers:** When the approximation consists not of reducing a set of correct answers to a smaller set, but when an individual answer is being approximated. For example, in determining the personal profile-category of a customer a more precise profile is better, but even a high level abstract profile-category is better than nothing. Another example is finding a product that satisfies as many requirements of a customer as possible. A product that only satisfies some of the requirements is already an approximate solution.
- **Disjointness of approximate and perfect answers:** Do the approximate answers come from the same domain as the perfect answers or not? In the customer-profile problem, no approximate abstract-class answer would ever qualify as the





perfect answer, since we always want a concrete (leaf) class as an answer (hence the domain of approximate solutions and of perfect solutions are disjoint), while in the product-selection problem, the approximate answer to one problem could be the perfect answer to another problem, hence the domain of approximate solutions and of perfect solutions coincide.

- **Two-sided approximation:** When both an upper bound and a lower bound are given, which approximate the perfect answer from two directions (e.g. both complete-unsound and sound-incomplete sets of answers).

## 6.3 Approximate Reasoning on the Semantic Web

The observation for introducing approximate reasoning to the Semantic Web field can be summarized as follows: most specification languages for ontologies are quite expressive, reasoning tasks are supposed to be very costly with respect to various resources, this being a crucial problem in the presence of large scale data.

For example, note that reasoning in most description logics that include general concept inclusion axioms (which is simply standard today, and e.g. the case in OWL DL) is at least EXPTIME complete, and if nominals are involved (as for OWL DL) even NEXPTIME complete. Although those worst case time complexities are not likely to be thoroughly relevant for the average behavior on real-life problems, this indicates that not every specifiable problem can be solved with moderate effort. These ideas of approximate reasoning are currently a cause of controversial discussions.

It is argued that soundness and completeness of Semantic Web reasoning is not to be sacrificed at all, in order to stay within the precise bounds of the specified formal semantics. On the other hand, it is also argued that the nature of many emerging Semantic Web applications involves data that is not necessarily entirely accurate, and at the same time is critical in terms of response time, so that sacrificing reasoning precision appears natural (Fensel & Harmelen, 2007[13]).

Another direction is to restrict knowledge representation to so-called tractable fragments that allow for fast, sound, and complete reasoning. Although this might be useful in scenarios where all essential knowledge can be modeled within the restricted fragment, in general there are strong arguments in favor of the usage of expressive formalisms. Firstly, real and comprehensive declarative modeling should be possible. A content expert wanting to describe a domain as comprehensive and as precisely as possible will not want to worry about limiting scalability or computability effects. Secondly, as research proceeds, more efficient reasoning algorithms might become available that could be able to deal with expressive specification formalisms more efficiently. Having elaborated specifications enables the reuse of knowledge in a more advanced way. Finally, elaborated knowledge specifications using expressive logics can reduce engineering effort by horizontal reuse: Knowledge bases could then be employed for different purposes because the knowledge is already there. However, if only shallow modeling is used, updates would require additional effort.

## 6.4 Further Studies

In this chapter, we give an introduction to what we have considered for LarKC on the evaluation of approximate reasoning, and we continue further studies of this topic in D1.4.1. We will precisely define approximate reasoning notions, which to date have



remained quite vague from the statistical perspective. We furthermore show that our mathematical modeling can be used for guiding the development of composed approximate reasoning systems. In the end, our mathematical modeling can be used for rigorous comparative statistical evaluation of approximate reasoning algorithms.



## 7. Evaluation of Rule-based Reasoners

This section describes the approaches which will be used to evaluate the performance and applicability of the rule based reasoner plugin to be introduced in more detail in D4.4.1 and implemented in D4.4.2. At the time of the writing of this deliverable an initial version of this plugin is already available, based on the IRIS [5] datalog engine. In general it is desirable to perform an evaluation of the reasoner plugin in terms of synthetic benchmarks as well as within LarKCs use-cases where possible. The advantages of synthetic benchmarks is that they are easily repeatable and allow an immediate comparison with existing systems based on already published and future results. Moreover, synthetic benchmarks can also highlight problematic corner cases, which might not occur very often in real-world situations but still hold very informative insights. On the other hand, an evaluation in terms of real use-cases and their associated datasets and reasoning requirements serves as a valuable assessment since generic, synthetic benchmarks might not capture a specific application area or might not reflect real-world requirements comprehensively. Thus an evaluation of the rule based reasoner plugin would ideally include both, synthetic benchmarks and a use-case centric evaluation.

Along with the type (synthetic benchmark or within a specific use-case), and thus the underlying motivation, of an evaluation of the rule-based reasoner plugin, it is also necessary to identify relevant performance metrics that should be collected.

Since we can use multiple custom rule-sets of varying complexity along with different reasoning strategies, we also have to take the inherent trade-off between reasoning efficiency, scalability and completeness of inference into account. Thus it is also crucial to include the quality of an answer as a metric, as pointed out in Section 3.3, and not regard other data-points in isolation. As pointed out, completeness means that a reasoner plugin can deduce all statements from a knowledgebase which are logically entailed by it. Incomplete reasoning can still infer valuable information and offer performance gains as a trade-off, however the completeness of query results and moreover the relevance of the results need to be considered.

Beyond that there are useful standard metrics which can be considered as relevant.

- **Load Time.** [44] identifies two main stages in ontology processing: Loading and pre-processing the data and query answering. Depending on the concrete evaluation strategy of the rule based reasoner plugin, reasoning might be performed by means of forward chaining at load time. Additionally parsing and indexing can make up significant quantity of loading time.
- **Query Time.** The second major task after reasoning is answering concrete SPARQL queries within a LarKC workflow. Depending on the underlying concrete evaluation strategy this step might involve reasoning, if a top-down evaluation algorithm (backward chaining) is used. Apart from that, several factors can have impact on the query response time, i.e. size and complexity of the dataset, the number of results, as well as the complexity of the query.

### 7.1 Synthetic Benchmarks

In terms of synthetic benchmarks there are several possible approaches to evaluating the rule based reasoners. At the very basic level it is possible to evaluate and compare a reasoner against other (Logic Programming based) rule engines. At a more focused



level it is also feasible to perform an evaluation of rule based reasoners as an RDF based inference plug-ins. In this case standard RDF/OWL benchmarks can be used. In the latter case, it becomes increasingly relevant to also consider SPARQL processing and query answering as a performance factor.

These two different approaches of performing an evaluation of the plugin both have their merits, however the results obtained each need to be examined in an appropriate context, because for RDF based query answering the relations are usually limited enough to allow more specific indexing.

So in order to get a broader picture of the performance of the rule based reasoner we will conduct evaluations in both directions, as an RDF based inference plugin, and as a general LP system. We acknowledge however that purely for the deployment within a LarKC workflow the first case delivers more indicative results.

### 7.1.1 Evaluation as a RDF based Reasoner

Wrapped as a LarKC plugin, it is straight-forward to evaluate the rule-based reasoner plugin as a generic RDF based inference engine, that performs RDF(S), *L2*, or a subset of OWL inference, and communicates as a SPARQL endpoint within the LarKC pipeline. This makes it possible to employ existing standard benchmarks, which has the advantage of facilitating easy comparison with other inference systems that explicitly target RDF or OWL based inference.

A well known example of an established synthetic benchmark focusing on (OWL based) inference is the Lehigh University Benchmark (LUBM) [18]. The data-sets used within the benchmark are generated by an application, which is part of the benchmark suite, according to user defined parameters. The produced data-sets consist of instance data over a fixed OWL ontology that describes a university organization. This makes it possible to construct repeatable experiments with standardized data-sets of a specific size.

Furthermore the LUBM benchmark also defines 14 queries based on which it is possible to evaluate the correctness and completeness of a system. Beyond this, the Berlin SPARQL Benchmark [6] or the SP2Bench test suite [35] are potentially relevant starting points for a more comprehensive and general evaluation which also includes and focuses on SPARQL processing.

### 7.1.2 Evaluation as a Rule Engine

Benchmarking directly at the level of a rule engine (a Logic Programming engine) is distinctively harder than performing a generic OWL/RDF benchmark. This is primarily for two reasons. Firstly, the lack of a standardized syntax makes it very hard to construct generic examples that could be used for the evaluation of different systems. Usually a translation step for each individual system is required to make even parsing possible (which depending on the size of the data-set, implemented optimizations, ...) itself can have impact on the performance of a particular system). Secondly, various systems support a wide range of different features, i.e. in terms of support for built-in predicates for arithmetic, implemented semantics (with the well-founded semantics [41] and stable model semantics [15] being the most common ones). These two points in combination make it hard to construct a general purpose benchmark that is applicable to a significant number of different systems and make it even hard gather meaningful results that allow a direct comparison of such systems.



A step towards solving this problem is OpenRuleBench [24]<sup>1</sup>. OpenRuleBench consists of a number of test frameworks, which makes it possible to run various benchmarks in a batch mode, collecting the results, and producing output. These different test cases stress particular aspects or features of rule engines. Due to the difficulties pointed out before, the rule sets for these test cases are hand crafted (and need adaptation for individual systems). However, the corresponding data-sets of varying sizes can then be created with data generators. The three main areas of focus for the test cases are (i) large join tests, (ii) recursion, and (iii) default negation.

IRIS, the rule engine used as basis of the rule based reasoner plugin, can already be evaluated within OpenRuleBench. We intend to gather more comprehensive performance figures and use those to further improve IRIS. As there are already results for several systems available, this will provide a very fine grained evaluation.

---

<sup>1</sup><http://rulebench.projects.semwebcentral.org/>



## 8. Evaluation of Unifying Search and Reasoning from the viewpoint of Granularity

Although the study of “Unifying Search and Reasoning from the viewpoint of Granularity” is an ongoing research topic. Many of its key strategies are potentially effective to LarKC use cases (This chapter assume the reader has read deliverable 4.3.1). Here we introduce some initial thoughts on the evaluation issue and how they are related with the use cases.

### 8.1 Quality of Service for Reasoning

In LarKC, according to the LarKC glossary, the Quality of Service (QoS) parameters are “provided by the end user or application issuing a query to a LarKC platform. It can be used to guide the way in which a particular workflow in the platform executes [1]”. Hence, the user context for a query is of vital importance for the quality of a reasoning result. The QoS can be evaluated from various aspects. Since unifying search and reasoning from the viewpoint of granularity emphasizes on removing the scalability barriers through satisfying the diversity needs from different users. The evaluation for each of the concrete strategies focuses on the quality of services. Here we mention some possible directions under the context of each concrete strategy designed for unifying search and reasoning from the viewpoint of granularity (More explanations on these strategies can be found in deliverable 4.3.1).

### 8.2 Evaluation of the Starting Point Strategy

The context for reasoning can be provided by the user or automatically acquired from user related events. In the starting point strategy, constraints (such as user interests) are automatically added to refine a SPARQL query. The evaluation can be performed as follows. Judging by the users, compared to the results based on user input query, if they are more satisfied with the query results acquired by the refined query, then reasoning with a starting point can get a relatively high score. If by using the starting point strategy, users have to be involved in the interleaving process for more times and spend more time compared to using the original query input by the users themselves, then reasoning with a starting point can get a relatively low score.

In deliverable 2.3.1, as a preliminary study of the starting point strategy for unifying search and reasoning, we have given an example on how to acquire users’ recent research interests from users’ publication lists and create a context for literature search. Our ongoing research uses the Medline dataset (provided by Ontotext) from WP7a to implement the same functionalities based on the SwetoDBLP dataset. We extract medical researchers’ recent interest based on their publication lists. Their recent interests are automatically added as additional constraints to their input query. As a joint collaboration, WP7a partner AstraZeneca has made some preliminary effort on collecting their scientists names for this experiment and we plan to invite these scientists to be involved in the evaluation of the refined query results.



### 8.3 Evaluation of the Multilevel Completeness Strategy

The multilevel completeness strategy is with anytime behavior, and it is designed as an interactive strategy which needs user involvement. When users choose to stop the reasoning process. At least two possible reasons should be considered.

- (1) Users are satisfied with the results that have been reasoned out.
- (2) As a compromise, users are satisfied with the current value of completeness.

For (1), it is very subjective and differs a lot according to different tasks, and users. For (2), a good prediction method and related evaluation for the value of completeness or specificity is necessary. In deliverable 4.3.1, we introduced a way of predicting the value of completeness. By correlation, we can evaluate if the prediction is good or not. Then users can judge whether the strategy and implementation is good enough for their use.

### 8.4 Evaluation of the Multiperspective Strategy

As introduced in deliverable 4.3.1, the perspectives for search and reasoning can be acquired through a starting point or suggested through an analysis of possible perspectives. In deliverable 4.3.1, we choose the predicates who have relatively higher value of edge degrees as recommended perspectives, whether this method is effective for general use needs further evaluation by the end users.

### 8.5 Evaluation Function for Combined Strategies

As introduced in deliverable 4.3.1, sometimes users may choose to combine several strategies together to produce results that they satisfy (such as the example of combining multilevel completeness strategy and the starting point strategy in Table 4.2). Hence, an evaluation method that has to consider multiple factors need to be developed. Here we provide an evaluation function, which is designed to be a linear weighted function. A weight is assigned on each of the factor considered, and each factor has a value that is provided by users according to their preference. The value of each factor shows how important of this factor does the user think. The evaluation function can be described as follows:

$$f(a, b, c, \dots, g, \dots) = w_1 * a + w_2 * b + w_3 * c + \dots + w_5 * g + \dots, \quad (8.1)$$

Where  $a, b, c, \dots, g, \dots$  serve as factors that affect the granular reasoning process and reasoning results. While  $w_1, w_2, \dots, w_n$  is the weight for each factor. And the effectiveness of combined strategies can be partially evaluated by this function.

In this chapter, we introduced some very preliminary design on the evaluation of “Unifying Search and Reasoning from the viewpoint of Granularity”. From the usecase perspective, they are going to be connected with WP7a dataset. More future studies will be done during year 2 of the LarKC project.



## 9. Analysis of Parallel Reasoning and Distributed Reasoning

### 9.1 Problem and motivation

Many application areas of LarKC involve reasoning or intelligently searching in very large search spaces. The reasoner is an important component of a workflow, as it determines to a large extent the workflows performance and scalability characteristics. Therefore, a reasoner can easily become a considerable bottleneck in the workflow's usability for use cases of real complexity. With regard to this, new innovative techniques for improving reasoning characteristics in a robust manner need to be elaborated in the LarKC project. In addition to a number of software solutions and implementation techniques enabled by LarKC, an infrastructure will be enabled apart from user desktop machines also large-scale high-performance (clusters of workstations, such as provided by the HLRS partner) and Grid/Cloud environment (both desktop and high-performance), ensuring that the full potential of progressive IT is enabled for the execution of plug-ins and workflows.

Due to a number of challenges posed for the efficient and scalable workflow's execution, we chose a reasoner as a pilot plug-in type for the investigation of techniques that ensure that the full potential of the LarKC infrastructure resources is enabled for the workflow execution. The experience gained and approaches elaborated for a reasoner will be adapted by other workflow's components.

For a reasoner, the realization of these techniques will allow distributed and parallel reasoning. Parallel reasoning is the main approach to speeding up the reasoning process using several computing nodes (processors) by running a reasoner on a parallel resource. Existing conceptions of achieving the combined benefits of using several computing nodes and intelligent reasoning will be adopted by LarKC. We will also concentrate on dealing with problems, such as balancing the processor loads or avoiding idle times, bottlenecks, and redundancy. On the scope of a workflow, parallel reasoning will be enabled by plug-in redistribution technique that allows execution of single plug-ins (in our case a reasoner) on remote resources, involving mentioned high-performance systems (distributed reasoning). The feedback will be given to the WP5 and used for improving the architecture's proposal for the LarKC project. Further, within the LarKC project the first analysis of adequate plug-ins takes place for a reasoner. For this the pellet reasoner (PelletSPARQLDLReasoner) is in the focus of the analysis for parallelization and distribution. However, before becoming concrete in terms of implementation models it is necessary to evaluate well proven distribution and parallelization techniques in order to meet the requirements of the LarKC project. The next sections will explain general approaches which are evaluated with the aim to adapt the most beneficial technique to a selected reasoner.

### 9.2 Conception

Several techniques are recognised for the implementation of parallel and distributed reasoning, including MPI, OpenMP, MapReduce, and others. In our investigations, we will analyse those techniques and elaborate a parallelisation and distribution strategy for several variations of reasoner.

The techniques are beneficial when one of the following applies for a reasoner's workflow (see Figure 9.1):



1. **Nested source code decomposition:** when source code's regions can be identified with clear control/data dependencies between them, for example in the form of a graph
2. **Nested data decomposition:** when data can be portioned and processed independently
3. **Inherited parallelism:** parallel execution of several processes at the same time.

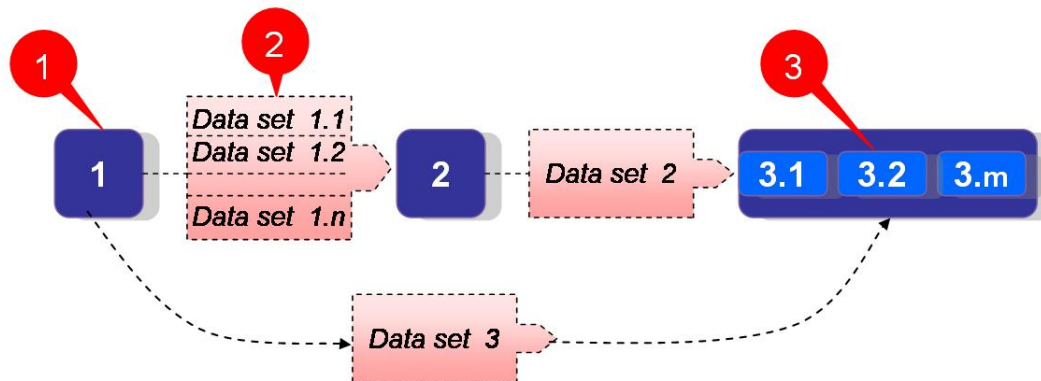


Figure 9.1: Some examples of nested parallelism in a reasoner's workflow

Based on code/data dependencies, several standard configurations of the workflow can be recognized for a reasoner that allows for parallel reasoning. Figures 9.2, 9.3 and 9.4 illustrate typical reasoner's workflows.

- **Single Code Multiple Data (SCMD workflow):** In this case the data that is being processed in the code region can be constructed of subsets that have no dependencies between them (see Figure 9.2). The same operation is performed on each of the subsets.
- **Multiple Code Single Data (MCSD workflow without conveyer dependencies):** For this workflow, several different operations are performed on the same data set. No dependencies between processed data set exist. This is typical for transformation of one data sets to another one according to rules that are specific for each subset of the produced data (see Figure 9.3)
- **Multiple Code Multiple Data (MCMD workflow):** This type of workflow is the combination of both previous workflows (SCMD and MSCD).

With regard to the current implementations in LarKC the mentioned techniques have to be considered. For this, we have to think not only of distribution and parallelization of one single plug-in but a whole workflow for executing reasoning task (e.g. combined with translation and selection tasks) should be considered. Nevertheless, the first step for adaption of such techniques is to use them for one single plug-in (distribution / parallelization of a reasoner). Further, we describe the distribution and parallelization strategies elaborated with regard to the identified workflow's types.

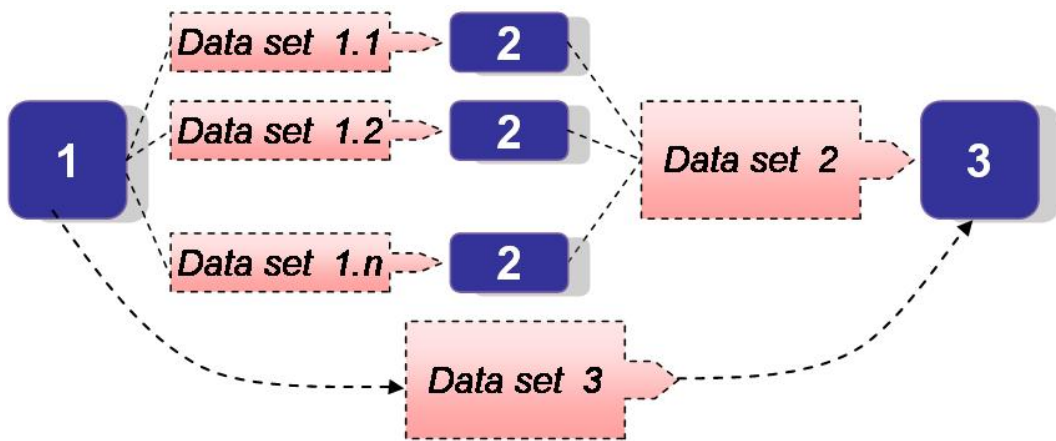


Figure 9.2: SCMD workflow

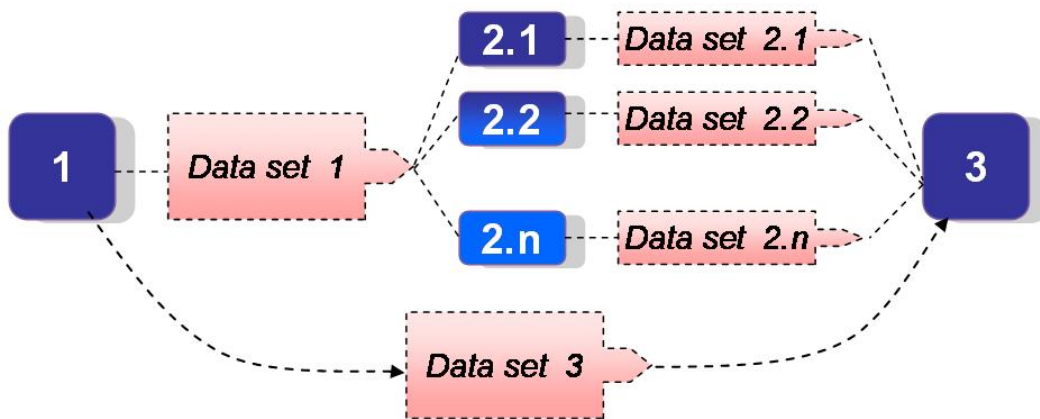


Figure 9.3: MCSD workflow

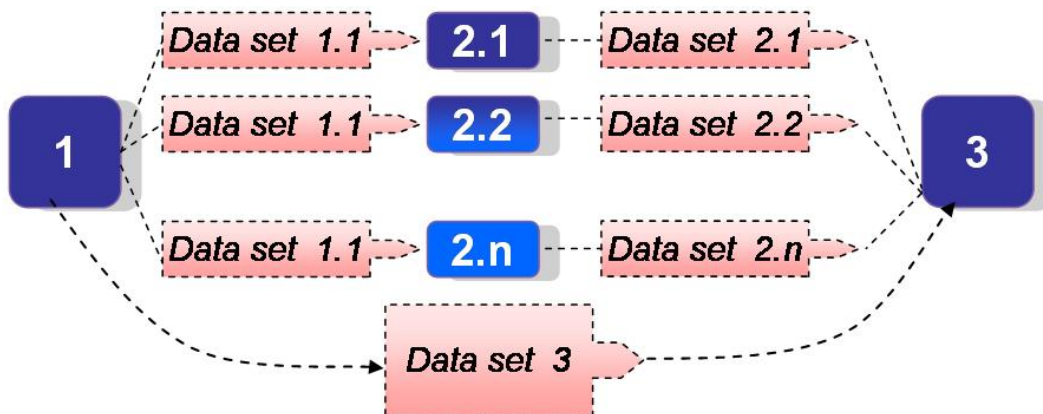


Figure 9.4: MCMD workflow



## 9.3 Technical solution and implementation

This section gives a short overview of the technologies that are the best practices from many years experience of LarKC partners in the field of parallel and distributed computing and can be potent used for WP developments. Describing the proposed solution, we highlight their usage scenarios in the LarKC project.

## 9.4 Distributed execution

When thinking of distributed execution there is a certain set of prerequisites to be considered. A distributed execution should be easy to implement, the support of several types of resources' system organisation (e.g. a Grid, a Cloud environment) should be considered and features for workflow monitoring and basic performance analysis should be considered as well.

- **Easy Implementation:** With regard to the current structure of the LarKC project, it is quite important to ensure a proper applicable way of implementation for a distributed execution in order to avoid a too complex structure. The utilization of distributed execution techniques for reasoning in LarKC is in the need of being simple in a way that enables developers to make use of the benefits without being challenged with complex problems. However, changes in the current code are required in order to adapt relevant techniques and increase performance and scalability.
- **Resource Types:** The usage of several resource types is required to ensure that all necessary resources are available und usable for a distributed execution for the reasoning methdos in LarKC. For this, techniques such as Grid Computing and Cloud Computing are considered because of efficiency and robustness.
- **Monitoring and Performance Analysis:** Methods for monitoring and basic applications for performance analysis ensure a tidy execution of the distributed execution for reasoning issues of LarKC. Monitoring techniques are required to ensure that during execution the reasoning processes are in line with the expectations. Furthermore, a performance analysis provides the developer with information for increasing scalability and hints motivates improvements to the current approach.

Identification of the software framework, that enables the requested features and allows for flexible components deployment on platforms from user the desktop to the supercomputer as a component of the Grid/Cloud, was done by WP4 in cooperation with WP5. As a result, the COMPs framework is proposed that is developed in the Barcelona Supercomputing Center and can greatly support distributed execution of the LarKC components.

COMPs [3] aims to facilitate the development of Grid applications, allowing for the inherent parallelism of LarKC components when running on the Grid. COMPs has three main distinctive features:

- The modular COMPs runtime follows the Grid Component Model (GCM), a component model especially designed for the Grid whose reference implementation is provided by ProActive realization.



- COMPs offers a straightforward programming model that particularly targets Java applications. The simplicity of this programming model keeps the Grid transparent to the user, who is able to program their applications in a Grid-unaware fashion. The user is only required to select the tasks to be run on the Grid, while the application can remain completely free of Grid-related calls.
- COMPs can use a wide range of Grid middlewares thanks to the JavaGAT API. JavaGAT provides a uniform interface for job submission and file transfer operations, being able to choose between different middlewares like Globus, UNICORE or SSH.

## 9.5 Parallel execution

For parallel execution there are some strong arguments towards use of the following approaches:

- **MapReduce:** MapReduce is a framework for the execution of processes running in parallel, it was invented by Google Inc. By usage of MapReduce the processes are executed on several nodes (e.g. sites in a grid environment). It is beneficial for the SCMD approach.
- **Message Passing Interface (MPI):** MPI [14] is a standard for describing the message exchange in distributed environments for parallel computations. MPI defines a set of operations for the processes that are executed in parallel. The Open MPI library will be used for the implementation of LarKC components, using the full MPI-2 standard implementation for the support of parallel applications. Both MCSD and MCMD approaches, which require an information exchange of the processes running in parallel, will benefit from the MPI realization by means of Open MPI.

Where possible, we will try to combine both of the approaches in order to reach the maximal productivity of the reasoner's parallel execution. For this, the reasoner itself has to be parallelized and if possible the processing of data will be done in parallel on distributed resources.



## 10. Testing Reasoners with Human Norm Data

### 10.1 Human Gold Standards

People are the ultimate judges of how well a reasoner handles everyday facts or even the output of a LarKC use case workflow. Unfortunately, the extent to which people agree with each other has to be taken into account when using human judgment as a standard. For example, human categorization judgments, even by experts, often do not agree well with each other (for a review, see [37]). As we will see, lay people rarely agree with each other 100% on anything but the most basic facts. There may well be higher consensus among people for the fact that ‘A screwdriver is-a tool’ than that ‘a knife is-a tool’. Such *low agreement* imposes an upper bound on how good a reasoner can be when measured against human judgment. What we need are measures on to what extent people agree with each other on basic facts. Such measures and datasets can be found in the psychological literature.

We distinguish between agreement through consensus and agreement through aggregation. While the former is common practice when developing ontologies, the latter is typically used in experimental psychology and knowledge elicitation. In agreement through consensus, people know they are working together. They can communicate with each other and monitor each other’s opinions as they form. They may even use tools that facilitate collaboration, such as ontology editing programs. Alternatively, In agreement through aggregation people make independent judgments without necessarily knowing that their responses will be later aggregated into published norms.

In section 10.2 we suggest how reasoners can be tested with crossvalidation using ontologies as data. Section 10.3 describes three sets of norms that are relevant for testing reasoners. In section 10.4 we summarize the main points and discuss how data from the two agreement integration methods could be combined.

### 10.2 Crossvalidation with ontologies

Ontologies, when designed by groups are expressions of human agreement. In cross validation one measures how well a model, parameterized on some training data, performs on future as-yet-unseen data. This is the approach taken to test the Inductive Materialization plug-in in WP3 (for more on the integration of machine learning in the semantic web, see deliverable 3.1. Coming from the statistics tradition as a measure to test generalization, cross validation can be applied to testing symbolic reasoning, where there is no parameter fitting to speak of, but there are models and data. In the context of cross validating reasoners, the model would be the reasoning mechanism. The ‘training’ data would be the majority of the ontology, and the ‘test’ dataset would be a small subset of the ontology that has been previously removed. This small test subset could vary in size, from one single statement to a sizeable percentage of the ontology. This procedure of reconstructing the excised parts could be iterated by randomly removing different parts; a reasoner’s average performance would be proportional to how well it can ‘rediscover’ the test data in all subsets.

We would expect reasoners and ontologies to interact, i.e., there would be no single reasoner that performs best on all possible knowledge domains. Therefore it is important to consider multiple ontologies. Typical general knowledge ontologies



include openCyc, WorldNet and DBpedia.<sup>1</sup> These are all part of the linked data project and heavily cross-reference each other, so in a way they work together instead of being mutually exclusive.

The *CyC* ontology contains “hundreds of thousands of terms, along with millions of assertions relating the terms to each other, forming an upper ontology whose domain is all of human consensus reality” [25] With version 2.0, the entire CyC ontology is now open source and probably the most well-developed upper ontology available, benefitting from 20 years of continuous development.

*WordNet* [29, 28] is a lexical database that brings together groups of synonyms (so-called synsets) in large networks, which show the semantic relationships between them. On the basis of the position of two words in such a network, it is possible to determine how different or similar the concepts are that they represent.

Miller and Fellbaum [28] used the strategy of deciding a priori diagnostic properties of the different domains, with different types of relational links for objects and events. In a way, it’s a lot more restricted than CyC and DBpedia in the relations it represents. For nouns referring to objects, relations such as synonymy, hyponymy (e.g., “dog” is a hyponym of “animal”), and meronymy (e.g., “mouth” is part of “face”) are argued to play a prime role in describing the semantic organization. For verbs, instead, the authors propose that relational links among verb concepts include troponymy (i.e., hierarchical relation in which the term at a level below, e.g., “crawling” is a manner of a term at a level above, e.g., travel/go/move/locomote), entailment (e.g., “snoring” entails “sleeping”) and antonymy (e.g., “coming” is the opposite of “going”), while relations such as meronymy would not apply. For cross validating a reasoner using WordNet, we could remove the proposition “coming is the opposite of going”, and then ask the reasoner what is the opposite of “going”.

*DBpedia* [7], which aspires to be an RDF version of Wikipedia, consists of around 274 million statements<sup>2</sup> scraped from the most structured part of Wikipedia, the infoboxes (which are small fraction of the total text of Wikipedia). DBpedia benefits from Wikipedia’s near real-time updating, so it reflects the current consensus about a particular topic. DBpedia thus represents real community agreement; it evolves along with Wikipedia.

Does DBpedia have normative value? Wikipedia seems to have established its reputation despite initial doubts, and assigning trust to Wikipedia content seems to be an active field. To some extent DBpedia inherits Wikipedia’s reputation, but to our knowledge there are no studies on DBpedia’s reliability or consistency. While using DBpedia it is easy to find minor annoyances such as multiple classes that represent the same concept and multiple equivalent relations; constructing a query takes some legwork, (e.g., visiting the pages for each candidate class.) Using DBpedia to cross validate reasoners may need to be postponed until DBpedia matures further.

## Limitations

Existing reference ontologies have some limitations. For a start, in *typical ontology* development other factors besides community *agreement* may come into play. Politics could be involved. (e.g., “should we put Volkswagen Passat in the luxury car category? The advertising department would like that”). Other subtler social factors could come into play as well.

<sup>1</sup>There are other general knowledge ontologies, such as YAGO and freebase, but they share enough features with the ontologies described here to be safely omitted

<sup>2</sup>According to <http://wiki.dbpedia.org/Datasets#h18-3>, visited 12/09/2009 10:42



Second, the rules for ontology development (and sometimes even the ontology editing tools) enforce consistency. That is, humans will produce more consistent ontologies under these conditions than when left to their own devices<sup>3</sup>.

Third, most ontologies are created by experts. This is reasonable; if we want to accurately capture a domain, we ask the experts. But ontologies may not reflect the knowledge of the lay person. There's also evidence that aggregated crowds' judgment can outperform experts', a phenomenon called 'the wisdom of crowds' [40].

The last problem is that there are no statistics on the degree of agreement for each statement. For DBpedia these could be inferred from editing history, but it would take quite a lot of extra assumptions and work. The number of edits on the Wikipedia page each triple came from could be a proxy for human agreement on the statement the triple represents. But there is no 1:1 match between a statement in Wikipedia and a tripple in DBpedia. Proxies such as number of edits *per pages* suffer from the credit assignment problem. For example, it could be that only a single statement in the page caused an edit war making all triples extracted from that page to be marked as controversial. Thus, it seems that currently there is no direct data on agreement for DBpedia triples.

Using norms instead of expert-created ontologies captures the average opinion. Norms are the result of people with no exceptional training. They are not created by a community, but typically by students that have no stake in the consistency or accuracy of the norms they will contribute to.

### 10.3 Crossvalidation with Human norms

This section describes what we called agreement through aggregation in the introduction, exemplified by psychological norms. Psychologists create norms mainly to generate better experiments. For example, in psycholinguistics, one may want to control for how related certain words are because they prime each other. In memory experiments, words that have similar features may be lumped together during study, and then retrieved together. Categorization experiments also need to control for previous knowledge people have. For these reasons, norming studies are useful and highly cited. The typical set up is that people get an open-ended question and they generate as many answers as possible (e.g., give members of the category 'furniture', or list features of 'chair'). Note that these norms differ from the consensual aggregation method (see previous section) in that people were not coordinated. That is, what one person wrote was not seen by the rest.

These norms are non-exhaustive: for example, if the object *accordion* has nine features in the dataset, that doesn't mean it cannot have more; it just means that the more popular features are those nine. All the norms we describe below use a threshold, i.e., they only include features if a minimum of say five people mentioned them.

Crossvalidation could be done on these features available on the dataset. For example, we could remove two out the nine features for *accordion*, and ask a reasoner to generate the missing features. The Inductive Materialization plug-in in WP3 does this kind of crossvalidation to infer features of people on a small FOAF dataset, but it uses statistical methods instead of a reasoner. There are some inference problems on structured data that both statistical methods and logic-based reasoners can be applied to (see deliverable D3.1 for more on this). Thus, testing techniques that are tried and true for statistical methods could be extrapolated to testing reasoners. The added

---

<sup>3</sup>although this hypothesis has not being tested to the best of our knowledge



advantage of using human norms is that for each statement we have an agreement value.

In general, category norms provide *is\_a* relations only. Feature norms should provide mainly *has\_a*, and *is\_a* relations, although other relations are possible (see examples in subsections below).

The common factor in the norming studies is that they describe how much agreement there is for each fact in the norms, and this distinguishes them from the previously mentioned ontologies. There are many varieties of norming studies (category norms, feature norms, plausibility norms and recall norms) but for simplicity here we discuss only the first two.

### 10.3.1 Category norms

Category norms simply relate objects to categories, representing the *is\_a* relation. We mentioned in the introduction that human judgments are often unreliable; category membership judgment is exceptional in that it has proved to be rather reliable, even across ages [21].

#### **Battig and Montague's (1969)**

The most popular norms are Battig and Montague's category norms [4]: Google scholar shows 1471 citations at the time of this writing. Due to their wide use, these norms were later recomputed with a more contemporary population [30]. Participants saw category names projected on a screen (see Table 1 for an Example) and they had to write as many items from that category as they could. After a limited time, a new category appeared and they had to list members again, and so on. The proportion of people that generated each item for each category is indicative of its typicality. Categories are overlapping, i.e., an item can be a member of more than one category.

As we can see in the example, typicality varies widely, which is one reason we should take it into account when dealing with *is\_a* statements.

**Limitations.** The categories we can access from the norms are but a small subset of the categories available in upper ontologies. Also, these norms do not include any kind of taxonomy that could help reasoning. In other words, in no way are we suggesting that norms should replace ontologies.

### 10.3.2 Feature norms

In feature norms people list as many features as they can think of for different objects. People are asked to provide a list of the features they believe to be important in describing and defining the meaning of a given word (See tables 2 and 3 for examples). Numerous statistics are available, such as feature saliency (production frequency, cue validity) and measures of how features are distributed across concepts. Both the McRae and Vinson's sets of norms are downloadable from [www.psychonomic.org/archive](http://www.psychonomic.org/archive).

#### **McRae et al.'s (2005)**

McRae et al. [26] collected feature norms from approximately 725 participants for 541 living (dog) and nonliving (chair) concepts. Each participant got a sheet with words that each denoted a concept. They needed to fill in as many of these lines as they



Table 10.1: : Example "furniture" and "reading materials" in the category norms

<b>An article of furniture</b>	<b>Proportion</b>
Chair(s)	0.9
Table	0.75
Couch	0.7
Bed	0.58
Desk(s)	0.49
Sofa	0.32
Dresser(s)	0.28
Loveseats(s)	0.26
Coffee-table	0.19
Lamp(s)	0.17
Nightstand(s)	0.13
Ottoman	0.1
Recliner	0.1
Stool(s)	0.11
end-table(s)	0.1
Futon	0.08
<b>A type of reading material</b>	<b>Proportion</b>
Magazines	0.93
Book(s)	0.92
Newspapers	0.71
Novel	0.26
Journal(s)	0.21
Article	0.17
Textbook(s)	0.17
Pamphlet(s)	0.13
Internet-related	0.09
Internet	0.05
website	0.03



could with properties of the concept to which the word refers. Table 2 shows examples of concepts and features from these norms.

Table 10.2: Two concepts, ACCORDION and AIRPLANE as example data from McRae et al (2005)

Concept	Feature	Production frequency	
ACCORDION	a_musical_instrument	28	
ACCORDION	associated_with_polkas	9	
ACCORDION	has_buttons	8	
ACCORDION	has_keys	17	
ACCORDION	inbeh_-_produces_music	6	
ACCORDION	is_loud	6	
ACCORDION	requires_air	11	
ACCORDION	used_by_moving_bellows	8	
ACCORDION	worn_on_chest	6	
AIRPLANE	beh_-_flies	25	
AIRPLANE	found_in_airport	8	8
AIRPLANE	has_a_propeller	5	
AIRPLANE	has_engines	5	
AIRPLANE	has_wings	20	
AIRPLANE	inbeh_-_crashes	7	
AIRPLANE	is_fast	11	
AIRPLANE	is_large	8	
AIRPLANE	made_of_metal	8	
AIRPLANE	requires_pilots	11	
AIRPLANE	used_for_passengers	15	
AIRPLANE	used_for_transportation	10	
AIRPLANE	used_for_travel	7	

McRae et al. norms contain very detailed information about features, for example how they relate to different sensory modalities. These ‘features of features’ were scored by independent judges.

**Limitations.** As with category norms, we do not have norms for all concepts. The bias here is towards concrete nouns; it is difficult to ask people for features of abstract concepts. Because people convey their conceptual knowledge through a linguistic filter, some types of information are transmitted more clearly than others. For example, information types such as parts (has a handle), color (is red), are easily verbalizable. However, other types of knowledge are omitted in verbal feature norms, particularly



when participants must produce features as short written descriptors. Events and verbs norms were added later on with Vinson and Vigglioco's norms.

### Vinson and Vigglioco (2008)

Vinson and Vigglioco went beyond the domain of nouns referring to objects, and applied the same techniques to the domain of actions and events. They tested 280 participants for a total of 456 words (169 nouns referring to objects, 71 nouns referring to events, and 216 verbs referring to events).

The speaker-generated features were classified into the five categories. First, *perceptual features*, defined as “features that describe information gained through sensory input, including body state and proprioception,” were identified and then subdivided into visual features, referring to the sense of vision (22.2% of all the features), and other perceptual features from other sensory modalities (19.7%). The nonperceptual features were then classified into *functional* (those features referring to the purpose of a thing, “what it is used for,” or the purpose or goal of an action; 26.5%), *motoric* (“how a thing is used, or how it moves,” or any feature describing such motor component of an action; 12.0%), and *other* (37.6%; the total percentage of scored features exceeds 100%, since some features met criteria for more than one feature type classification).

The main advantage of this dataset over McRae et al.'s is that different meanings are evaluated separately. McRae attempted to avoid ambiguous concept names, although this is impossible given the fact that such a large proportion of English words are ambiguous. When a word is both a verb and a noun, Vinson and Vigglioco asked people to generate features for the noun and for the verb meanings separately (e.g., hammer as an object, rather than the verb meaning of hammer). A second, minor advantage is that features are often one-word (see table 3).

**Limitations.** As before, feature norms deal well with concrete concepts, but not so well with abstract ones (What are the features of commitment?). Words referring to events are more abstract than words referring to objects, so in that sense Vinson and Vigglioco's [43] norms are a good step in the right direction.

A common limitation to all feature norms is that information is in the form: subject + predicate (that is, object + feature). For maximum consistency with classical triple structure, we would need to parse the predicate into a verb and an object. This may not be a trivial endeavor for some features. For example, In table 3, ‘in ARTICHOKE *is* round’ the verb ‘is’ would be a reasonable inference, but other features such as *digest*, *water*, and *health* are harder to rewrite as a verb and an object.

An additional common limitation of human norms is that they are much reduced in scope when compared to mega-ontologies such as CyC, DBpedia and WordNet. However, nothing prevents us from collecting additional norming data, as a specific need arises (for example, in the use cases).

## 10.4 Discussion

This section presents two ideas:

(1) It is possible to use existing cognitive science data (norms) as a technique to evaluate reasoning with everyday facts. With norms, we have human agreement data for every single statement. When testing reasoners, missing a low-agreement fact should not be penalized as much as missing a high-agreement one.

Furthermore, linking norms to general knowledge ontologies would combine the strengths of both methods. For example, thanks to category norms (see Table 1) we



Table 10.3: Two nouns (ARTICHOKE, HORSE), two verbs (ARGUE, STOP) from the Vinson and Vigglioco (2008) feature norms

Feature	ARTICHOKE	feature	HORSE	Feature	ARGUE	feature	STOP
Vegetable	18	animal	18	communicate	6	action	11
Green	13	ride	12	Fight	5	motionless	7
Eat	10	4-legs	10	Humans	4	sign	6
Heart	6	leg	8	Yell	4	cease	6
Health	6	mammal	7	Voice	4	end	5
Round	4	mane	6	Opinion	3	move	4
Cook	3	fast	6	2	2	humans	2
Flower	2	humans	5	Word	2	intentional	2
Digest	2	beautiful	4	Loud	2	finish	2
Leaf	2	tail	4	Action	2	involuntary	1
Water	2	transport	4	stubborn	1	object	1
Plant	2	strong	4	attempt	1	pause	1
Grow	2	hoof	3	conversation	1	friction	1
Soft	2	race	3	Explain	1	brake	1
Object	2	farm	3	Idea	1	Do	1
Humans	2	big	2	Point	1	location	1
Desert	1	hair	2	Stand	1		
Purple	1	diff- colors	2	Lose	1		
Consume	1	large	2	Scare	1		
Boil	1	work	2	Upset	1		
Salad	1	water	1	Process	1		
Peel	1	track	1	Mean	1		
Fat	1	bit	1	Talk	1		
Ingredient	1	grace	1	Violent	1		
Crunch	1	compete	1	Hit	1		
Nature	1	domestic	1	Speak	1		
Garden	1	sleep	1	Verbal	1		
Tree	1	coat	1	Anger	1		
White	1	shout	1	Verb	1		
Taste	1	back	1	emotion	1		
		Eats	1	Bad	1		
		Friendly	1	involuntary	1		
		War	1	intentional	1		
		Show	1				
		Head	1				
		Ear	1				
		Sport	1				
		Eye	1				
		Walk	1				



know that a chair is considered an article of furniture more strongly than a sofa; this information could be added to an ontology. WP3 is working on adding an extension to SPARQL that would consider probabilities (the `PROB` modifier), and the storage layer in WP5 allows for a fourth slot for each triple that could well contain this information. While merging the norm and ontology data perhaps could improve reasoning with everyday facts, it is not directly related to testing reasoners, so this idea would be explored somewhere else.

(2) Crossvalidation methods that are common in machine learning and cognitive science may well be applicable to testing logic-based reasoners. While there are many attempts in the literature to combine statistical and logic-based reasoning (e.g., [23, 11]), there seems to exist less work on testing both approaches with the same methods and data. Crossvalidation seems like a sensible first step and could open the door to more thorough exchange of testing methods.



## 11. Conclusion

This document is designed to present an initial framework of evaluation and benchmarking of reasoners for the LarKC platform. Therefore, what we have done in this document is to define the evaluation methods, measures, benchmarks, and performance targets for the plug-ins to be developed in various tasks such as approximate reasoning with interleaved reasoning and selection and rule-based reasoning. Those measures are examined with the perspectives of the use-cases of WP6 and WP7, as well as in terms of synthetic benchmarks.

In the subsequent document of this deliverable D4.7.2, we will perform experiments on reasoners with the data sets, and report our evaluation and benchmarking of those reasoners. Moreover, we will evaluate the reasoners by using other benchmarks developed in other projects. The evaluation methods and benchmarks developed in this document will be integrated with the SEALS Platform.

SEALS <sup>1</sup> is a project on Semantic Evaluation at Large Scale. The goal of the SEALS project is to provide an independent, open, scalable, extensible, and sustainable evaluation infrastructure for semantic technologies. The SEALS Platform allows the remote evaluation of semantic technologies thereby providing an objective comparison of the different existing semantic technologies. This will allow researchers and users to effectively compare the available technologies, helping them to select appropriate technologies and advancing the state of the art through continuous evaluation. The SEALS Platform will provide an integrated set of semantic technology evaluation services and test suites. They will be used in two public and worldwide evaluation campaigns. The results of these evaluation campaigns will be used to create semantic technology roadmaps identifying sets of efficient and compatible tools for developing large-scale semantic applications. It would be very useful to integrate the evaluation methods and benchmarks developed in the context of LarKC with the SEALS Platform.

---

<sup>1</sup><http://www.seals-project.eu/>



## REFERENCES

- [1] Larkc glossary, 2009.
- [2] Bo Andersson and Vassil Momtchev. D7a.1.1 - requirements summary and data repository, September 2008. Available from: <http://www.larkc.eu/deliverables/>.
- [3] BSC Barcelona Supercomputing Center. Web page of the comp superscalar framework, 2009. Available at the BSC page at [http://www.bsc.es/plantillaG.php?cat\\_id=547](http://www.bsc.es/plantillaG.php?cat_id=547).
- [4] W. F. Battig and W. E. Montague. Category norms for verbal items in 56 categories - a replication and extension of connecticut category norms. *J Exp Psychol*, 80(3P2):1–8, 1969.
- [5] B. Bishop and F. Fischer. IRIS-Integrated Rule Inference System. In *Proceedings of the 1st Workshop on Advancing Reasoning on the Web: Scalability and Commonsense (ARea2008) hosted by the 5th European Semantic Web Conference (ESWC-08)*, 2008.
- [6] C. Bizer and A. Schultz. Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints, 2008.
- [7] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Science, services and agents on the world wide web : DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 2009.
- [8] Irene Celino, Daniele Dell’Aglío, Emanuele Della Valle, and Kono Kim. D6.3 - Urban Computing environment specification, May 2009. Available from: <http://www.larkc.eu/deliverables/>.
- [9] Mukesh Dalal. Anytime clausal reasoning. *Annals of Mathematics and Artificial Intelligence*, 22(3–4):297–318, 1998.
- [10] Daniele Dell’Aglío, Emanuele Della Valle, and Irene Celino. D6.4 - 2nd periodic report on data and performances, September 2009. Available from: <http://www.larkc.eu/deliverables/>.
- [11] Y. Ding. IR and AI: the role of ontology. In *Proceedings of 4th International Conference of Asian Digital Libraries, Bangalore, India*, page 10–12, 2001.
- [12] W. P. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1:267–284, 1984.
- [13] Dieter Fensel and Frank Van Harmelen. Unifying reasoning and search to web scale. *IEEE Internet Computing*, 11(2):94–96, March/April 2007.
- [14] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard, June 2008. Published by High-Performance Computing Center Stuttgart, HLRS.
- [15] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080, 1988.



- [16] P. Groot, A. ten Teije, and F. van Harmelen. Towards a structured analysis of approximate problem solving: a case study in classification. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR'04)*, Whistler, Colorado, June 2004.
- [17] Perry Groot, Heiner Stuckenschmidt, and Holger Wache. Approximating description logic classification for semantic web reasoning. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*, volume 3532 of *Lecture Notes in Computer Science*, pages 318–332. Springer, 2005.
- [18] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):158–182, 2005.
- [19] Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The Instance Store: DL reasoning with large numbers of individuals. In *Proceedings of the International Workshop on Description Logics, DL2004*, Whistler, Canada, pages 31–40, 2004.
- [20] E. J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In L. N. Kanal, T. S. Levitt, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, pages 301–324. Elsevier, Amsterdam, The Netherlands, 1987.
- [21] D. V. Howard. Category norms: A comparison of the battig and montague (1969) norms with the responses of adults between the ages of 20 and 80. *The Journal of Gerontology*, 35(2):225, 1980.
- [22] Zhisheng Huang, Johanna Volker, Qiu Ji, Heiner Stuckenschmidt, Christian Meilicke, Stefan Schlobach, Frank van Harmelen, and Joey Lam. Benchmarking the processing of inconsistent ontologies, knowledgeweb d1.2.2.1.4/d2.1.6.3, 2007. Available from: <http://wasp.cs.vu.nl/knowledgeweb/D2163/kweb2163.pdf>.
- [23] K. S. Jones. Information retrieval and artificial intelligence. *Artificial Intelligence*, 114(1):257–281, 1999.
- [24] S. Liang, P. Fodor, H. Wan, and M. Kifer. OpenRuleBench: an analysis of the performance of rule engines. In *Proceedings of the 18th international conference on World wide web*, pages 601–610. ACM New York, NY, USA, 2009.
- [25] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira. An introduction to the syntax and content of cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, page 44D49, 2006.
- [26] K. McRae, G. S. Cree, M. S. Seidenberg, and C. McNorgan. Semantic feature production norms for a large set of living and nonliving things. *Behavioral Research Methods, Instruments, and Computers*, 37:547D559, 2005.
- [27] Alistair Miles and Sean Bechhofer, August 2009. Available from: <http://www.larkc.eu/deliverables/>.





- [28] G. A. Miller and C. Fellbaum. Semantic networks of english. *Cognition*, 41(1-3):197, 1991.
- [29] GA Miller, R Beckwith, C Fellbaum, D Gross, and KJ Miller. Introduction to WordNet: an on-line lexical database\*. *International Journal of Lexicography*, 3(4):235–244, 1990.
- [30] James Van Overschelde, Katherine Rawson, and John Dunlosky. Category norms: An updated and expanded version of the norms. *Journal of Memory and Language*, 50(3):335, 289, 2004.
- [31] Jeff Z. Pan and Edward Thomas. Approximating OWL-DL ontologies. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1434–1439. AAAI Press, 2007.
- [32] Angus Roberts, Kurt Straiß, James McKay, Martin Stetter, and Hamish Cunningham. D7b.1.1a - requirements summary and data repository, November 2008. Available from: <http://www.larkc.eu/deliverables/>.
- [33] Sebastian Rudolph, Tuvshintur Tserendorj, and Pascal Hitzler. What is approximate reasoning? In *Proceedings of RR2008, LNCS 5341*, pages 150–164, 2008.
- [34] Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74:249–310, 1995.
- [35] Michael Schmidt, Thomas Hornung, Georg Lausen, and Christoph Pinkel. Sp2bench: A sparql performance benchmark. *CoRR*, abs/0806.4627, 2008. informal publication.
- [36] B. Selman and H. A. Kautz. Knowledge compilation using Horn approximations. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 904–909, 1991.
- [37] J. Shanteau, E. Salas, and G. Klein. What does it mean when experts disagree? In *Linking expertise and naturalistic decision making*, pages 229–244. Lawrence Erlbaum Associates, Mahwah, NJ, 2001.
- [38] Heiner Stuckenschmidt. Partial matchmaking using approximate subsumption. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1459–1464. AAAI Press, 2007.
- [39] Heiner Stuckenschmidt and Frank van Harmelen. Approximating terminological queries. In H.L. Larsen and et al, editors, *Proc. of the 4th International Conference on Flexible Query Answering Systems (FQAS)'02*, Advances in Soft Computing. Springer, 2002.
- [40] James Surowiecki. *The Wisdom of Crowds*. Anchor, August 2005.
- [41] A. VAN GELDER, K.A. ROSS, and J.S. SCHLIPF. The Well-Founded Semantics for General Logic Programs. *Journal of the Association for Computing Machinery*, 38(3):620–650, 1991.



- [42] F. van Harmelen and A. ten Teije. Describing problem solving methods using anytime performance profiles. In *Proceedings of ECAI'00*, pages 181–186, Berlin, August 2000.
- [43] D. P. Vinson and G. Vigliocco. Semantic feature production norms for a large set of objects and events. *Behavior Research Methods*, 40(1):183, 2008.
- [44] T. Weithoner, T. Liebig, M. Luther, and S. Bohm. What's Wrong with OWL Benchmarks? In *Proceedings of the Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006)*, pages 101–114, 2006.
- [45] Michael Witbrock, Blaz Fortuna, Luka Bradesko, Mick Kerrigan, Barry Bishop, Frank van Harmelen, Annete ten Teije, Eyal Oren, Vassil Momtchev, Axel Tenschert, Alexey Cheptsov, Sabine Roller, and Georgina Gallizo. D5.3.1 - requirements analysis and report on lessons learned during prototyping, Joune 2009. Available from: <http://www.larkc.eu/deliverables/>.