NeOn-project.org

NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 — "Semantic-based knowledge and content systems"

D6.8.2 Testing the NeOn Toolkit interoperability

Deliverable Author: **Raúl García-Castro**

Deliverable Authoring Institution: **Universidad Politécnica de Madrid (UPM)**

This deliverable includes the analysis of the results of the second iteration of benchmarking the interoperability of the NeOn Toolkit using RDF(S) and OWL as interchange languages, as well as the recommendations extracted from this analysis for improving the NeOn Toolkit.

Document Identifier:	NEON/2008/D6.8.2/v1.1	Date due:	November 30, 2008
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	November 30, 2008
Project start date	March 1, 2006	Version:	v1.1
Project duration:	4 years	State:	Final version
		Distribution:	Public

NeOn Consortium

This document is part of the NeOn research project funded by the IST Programme of the Commission of the European Communities, grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta}@open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB D-76128 Karlsruhe Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.ump.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail adress: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SL–1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 665 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier France Contact person: Jérôme Euzenat	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield United Kingdom Contact person: Hamish Cunningham
Universität Kolenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Marino della Battaglia 44 – 00185 Roma-Lazio Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document:

- Universidad Politécnica de Madrid
- Ontoprise

Change Log

Version	Date	Amended by	Changes
0.1	15-09-2008	Raúl García-Castro	First draft of the document
0.2	27-10-2008	Raúl García-Castro	RDF(S) and OWL interoperability benchmarking descriptions
0.3	10-11-2008	Raúl García-Castro	RDF(S) interoperability results
0.4	14-11-2008	Raúl García-Castro	OWL interoperability results
0.4	17-11-2008	Raúl García-Castro	Evolution of results
0.5	20-11-2008	Raúl García-Castro	Extension of the evaluation
1.0	21-11-2008	Raúl García-Castro	First version
1.1	10-12-2008	Raúl García-Castro	Implemented the comments from the quality reviewer (Michael Erdmann)

Executive Summary

The Knowledge Web European Network of Excellence has organised two benchmarking activities with the goal of assessing and improving the interoperability of Semantic Web technology using RDF(S) and OWL (the languages recommended by the W3C) as interchange languages.

Within the NeOn project, the NeOn Toolkit has been benchmarked with the methods and benchmark suites provided for these benchmarking activities. A first round of the interoperability results was presented in 2007 and, this year, we have re-evaluated the NeOn Toolkit with the latest version available at the time of writing this deliverable, which includes OWL support.

The results show no problems regarding RDF(S) interoperability but, although the interoperability of the NeOn Toolkit has significantly improved since it has native OWL support, the NeOn Toolkit is not a full OWL-interoperable tool because of its behaviour when dealing with ontologies that include blank nodes and ontologies that include properties with a range of *rdfs:Literal* or a XML Schema datatype.

These issues do not prevent a correct working of the NeOn Toolkit. However, they hinder its interoperability with other tools and its adoption by a broad range of users. Therefore, we provide a set of recommendations to improve the interoperability of the NeOn Toolkit.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Knowledge Web[1] European Network of Excellence organised two benchmarking activities with the goal of assessing and improving the interoperability of the Semantic Web technology using an interchange language; the interchange languages used for those activities are those recommended by the W3C, namely RDF(S) and OWL.

In the NeOn project, the interoperability of the NeOn Toolkit has been evaluated with the methods and benchmark suites provided for these benchmarking activities. Therefore, and with the aim of obtaining objective evaluation results, the whole evaluation and the analysis of the results have been performed by a person who does not belong to the NeOn Toolkit developers group.

A first round of the interoperability results was presented in 2007 in the NeOn deliverable D6.8.1[GC07], including the results of version 1.0 B823 of the NeOn Toolkit, which run in the Frame-Logic mode with no native OWL support. This year, we have re-evaluated the NeOn Toolkit with the latest version available at the time of writing this deliverable (1.2.0 B739), which includes native OWL support.

This deliverable includes the analysis of the results of benchmarking the interoperability of the NeOn Toolkit with other Semantic Web tools using RDF(S) and OWL as interchange languages; it also includes the recommendations extracted from the analysis performed for improving the NeOn Toolkit.

When benchmarking the interoperability using RDF(S), the NeOn Toolkit was evaluated with three tools: Jena, WebODE, and itself, whereas when benchmarking the interoperability using OWL, the NeOn Toolkit was evaluated with nine tools: GATE, Jena, KAON2, Protégé Frames, Protégé OWL, SemTalk, SWI-Prolog, WebODE, and itself.

Since we have evaluated the interoperability of the NeOn Toolkit twice, in this deliverable we also present the evolution over time of the interoperability results of the NeOn Toolkit. In order to clearly identify the improvement (or loss) of interoperability gained after updating the NeOn Toolkit, we chose to perform the experiments maintaining the versions of the tools used in the previous experiments.

We expected not to find any execution failure in the tool and to correctly interchange with the other tools the common parts of their knowledge models. It must be noted that interoperability also depends on the other tools participating in the interchanges.

This deliverable is structured as follows. Chapter 2 gives a description of the RDF(S) and OWL interoperability benchmarking activities, of the experiment defined in these activities and of the updates performed for this second evaluation of the NeOn Toolkit. Chapters 3 and 4 present the analysis performed on the NeOn Toolkit interoperability results. Chapter 5 offers some recommendations to improve the interoperability of the NeOn Toolkit and, finally, chapter 6 draws some conclusions from the results.

[1] `http://knowledgeweb.semanticweb.org/`

Chapter 2

Interoperability benchmarking

As commented above, the Knowledge Web[1] European Network of Excellence organised two benchmarking activities, the RDF(S) Interoperability Benchmarking [GCGPS07] and the OWL Interoperability Benchmarking [GCGP08], that had two main goals[2]:

- **To assess and improve the interoperability of Semantic Web technologies** using RDF(S) and OWL as interchange languages. This would permit learning about the current interoperability of the tools and maximising the knowledge that these tools can interchange while minimising the information addition or loss.
- **To identify the fragment of knowledge that the Semantic Web technologies can share** using RDF(S) and OWL as interchange languages. As this fragment becomes larger, more expressive ontologies can be interchanged among these technologies.

These two benchmarking activities followed the Knowledge Web benchmarking methodology [GCMW+04] for Semantic Web technologies and provided the following resources for automatically evaluating the interoperability of Semantic Web technologies:

- A manual and an automatic experimentation approach for benchmarking interoperability. For benchmarking the interoperability of the NeOn Toolkit we have followed the automatic experimentation approach described in section 2.1.
- Several ontology datasets for evaluating the import, export and interoperability capabilities of the tools that contain ontologies with simple combinations of the RDF(S) and OWL knowledge models. The ontology dataset used for benchmarking the RDF(S) interoperability of the NeOn Toolkit has been the RDF(S) Import Benchmark Suite[3] and the ontology dataset used for benchmarking the OWL interoperability has been OWL Lite Import Benchmark Suite[4].
- The IBSE (Interoperability Benchmark Suite Executor) tool[5], which is the interoperability evaluation infrastructure that automates the execution of the experiments and provides HTML summarised views of the obtained results.

In the previous iteration of the interoperability benchmarking we detected some problems with the tool that was being used for comparing OWL ontologies. Therefore, as described in section 2.2, in this iteration we changed the OWL ontology comparer to avoid these problems.

[1] http://knowledgeweb.semanticweb.org/
[2] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/
[3] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/rdfs/rdfs_import_benchmark_suite.html
[4] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/owl/import.html
[5] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/ibse

Furthermore, as the ontology dataset used in the previous iteration only contained OWL Lite ontologies, we have extended the evaluation to cover real-world ontologies. The new ontologies included in the interoperability evaluation are described in section 2.3.

2.1 Experiment performed

The experiment performed consisted of measuring the interoperability of the tools through the interchange of ontologies from one tool to another. From these measurements, we can extract the interoperability between the tools, the causes of problems, and improvement recommendations.

Of the different ways that Semantic Web tools have to interoperate, we only consider interoperability when the tools interchange ontologies by using an interchange language. Therefore, the functionalities affecting the results are the importers and exporters of the tools to the interchange language. Besides, with no human intervention, we can only access tools through application programming interfaces (APIs), and thus the operations performed to access them must be supported by most of the Semantic Web tools. Therefore, the only operations to be performed by a tool should be the following: to import one ontology from a file (to read one file with an ontology and to store this ontology in the tool knowledge model), and to export one ontology into a file (to write an ontology stored in the tool knowledge model into a file).

During the experiment, a common group of benchmarks is executed and each benchmark describes one input ontology that has to be interchanged between a single tool and the others.

Each benchmark execution comprises two sequential steps, shown in Figure 2.1. Starting with a file that contains an ontology, the first step (*Step 1*) consists in importing the file with the ontology into the original tool and then exporting such ontology into a file using the interchange language. The second step (*Step 2*) consists in importing the file with the ontology (exported by the original tool) into the destination tool and then exporting such ontology into another file.

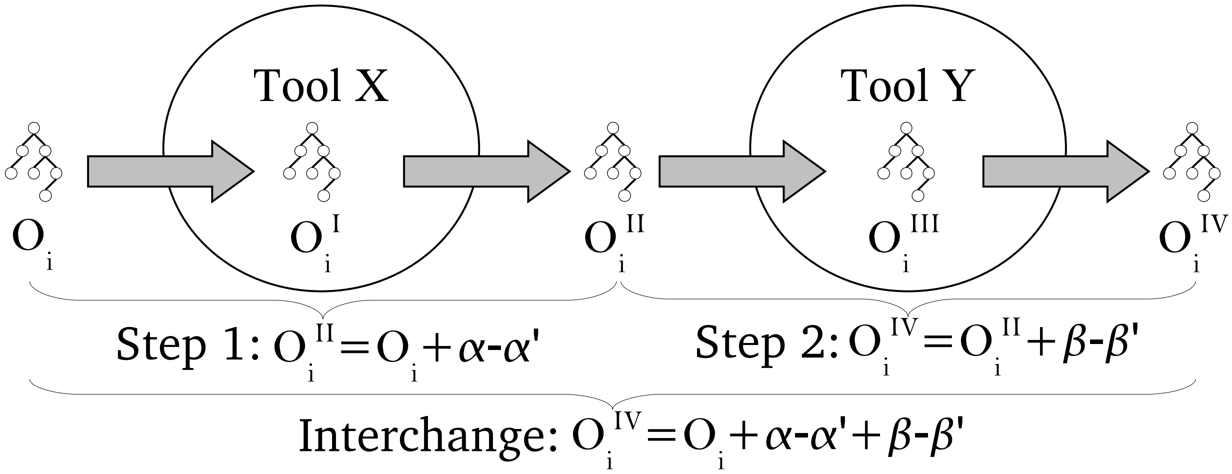

Figure 2.1: The two steps of a benchmark execution.

In these steps, there is not a common way for the tools to check the results of importing the ontologies, we just have the results of combining the import and export operations (the files exported by the tools), so we consider these two operations as an atomic operation. It must be noted, therefore, that if a problem arises in one of these steps, we cannot know whether the problem was originated when importing or when exporting the ontology since we do not know the state of the ontology inside each tool.

After a benchmark execution, the results obtained from the ontology described in the benchmark are three different states, namely, the original ontology, the intermediate ontology exported by the first tool, and the final ontology exported by the second tool. From these results we define the evaluation criteria for a benchmark execution. These evaluation criteria will be considered in *Step 1*, *Step 2*, and in the whole interchange (*Step*

1 + *Step 2*); they are the following:

- **Execution** (*OK*/*FAIL*/*C.E.*/*N.E.*) informs of the correct execution of a step or the whole interchange. Its value is *OK* if the step or the whole interchange is carried out with no execution problem; *FAIL* if the step or the whole interchange is carried out with some execution problem; *C.E.* (Comparer Error) if the comparer launches an exception when comparing the original and the final ontology; and *N.E.* (Not Executed) if the second step is not executed because the execution on the first step failed.
- **Information added or lost** informs of the information added to or lost from the ontology in terms of triples in each step or in the whole interchange. We can know the triples added or lost in *Step 1*, in *Step 2*, and in the whole interchange by comparing the original ontology with the intermediate one, then the intermediate ontology with the final one, and the original with the final ontology, respectively.
- **Interchange** (*SAME*/*DIFFERENT*/*NO*) informs whether the ontology has been interchanged correctly with no addition or loss of information. From the previous basic measurements we can define *Interchange* as a derived measurement that is *SAME* if *Execution* is *OK* and *Information added* and *Information lost* are void; *DIFFERENT* if *Execution* is *OK* but *Information added* or *Information lost* are not void; and *NO* if *Execution* is *FAIL*, *N.E.* or *C.E.*.

2.2 Changes in the OWL ontology comparer

One of the conclusions drawn from the previous evaluations was that we needed to change the OWL ontology comparer, because it did not cover our requirements for such a comparer. We need a tool that, first, poses no execution problems and, second, automatically compares OWL ontologies from the lexical to the conceptual (also known as semantic) level.

Figure 2.2 shows the different ontology heterogeneity levels (see [Bar07] for a comparison of the different approaches found in the literature). We do not aim to compare OWL ontologies at the pragmatic level, which consists in encountering all the discrepancies that result from the fact that different individuals/communities may interpret the same ontology in different ways in different contexts; finding these differences automatically is not possible because it requires a human to interpret the ontologies.

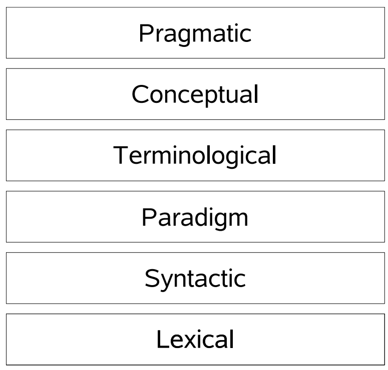

Figure 2.2: Ontology heterogeneity levels.

The KAON2 OWL Tools[6], the OWL ontology comparer used in the previous version of the IBSE tool, has some problems. In some cases it had execution problems when processing ontologies (e.g., with ontologies that are not syntactically correct) and in other cases the output produced was not correct. In this second case, sometimes it was because the KAON2 OWL Tools is based in KAON2, a non-native OWL tool.

After researching on other existing tools, we found no tool able of comparing two OWL ontologies and of providing the differences between them, as required for our case. Therefore, we aimed for a combined solution using several tools.

[6] `http://owltools.ontoware.org/`

We decided to use Jena[7] and Pellet[8] because they are freely available[9] and because they provide well documented programming interfaces. Moreover, we had to use both Jena and Pellet because in some cases they do not compare OWL ontologies correctly and in punctual cases they have execution problems.

This way, Pellet provides a comparison at the semantic level and Jena provides a comparison at the structural (isomorphic) level that is faster than the previous one, syntax-checking capabilities, and the ability of extracting the differences between two ontologies.

The process followed for comparing two OWL ontologies (*O1* and *O2*) combining Jena and Pellet is described below. We can see that it is a process that depends on the deficiencies of the tools for performing the required ontologies.

1. To check with Jena that the ontologies are syntactically valid.
2. To check with Jena if the ontologies are isomorphic. If *O1* is isomorphic with *O2*, they are structurally identical and, therefore, equivalent.
3. To check with Pellet if the ontologies are entailed by the other. If *O1* entails *O2* and *O2* entails *O1*, they are equivalent.
4. To extract with Jena the differences between the ontologies (*diff1* and *diff2*). Jena extracts these differences as RDF graphs.
5. To check with Jena if the graphs with the differences between the ontologies are isomorphic. If *diff1* is isomorphic with *diff2*, the ontologies are equivalent.

During the process, we also check for the following issues when the tools compare ontologies:

- In some cases, Jena and Pellet say that one literal value and the same literal value with a *xsd:string* datatype (e.g., "Peter" and "Peter"^^<xsd:string>) are different. But according to the RDF Datatype entailment rules[10], these literals are equivalent.
- Jena considers that two ontologies named "onto" and "onto#" are different.
- Pellet considers that two ontologies entail each other when one has the "*ontName rdf:type owl:Ontology.*" triple and the other does not have it.

2.3 Extension of the interoperability evaluation

In order to extend the evaluation of the interoperability of the NeOn Toolkit, covering more cases than those present in the OWL Lite Import Benchmark Suite, we asked the NeOn partners to provide real-world ontologies to be used in the interoperability evaluations.

Three different groups of ontologies were provided:

- Three ontologies from the FAO Agricultural Information Management Standards: *aos.owl*[11], *asc.owl*[12] and *languagecode.owl*[13].

[7] http://jena.sourceforge.net/
[8] http://pellet.owldl.com/
[9] Jena is open source and Pellet may be used in open source applications under the terms of the AGPL version 3 license.
[10] http://www.w3.org/TR/rdf-mt/#DtypeRules
[11] http://www.fao.org/aims/aos/aos.owl
[12] http://www.fao.org/aims/aos/asc/asc.owl
[13] http://www.fao.org/aims/aos/languagecode.owl

- The seven ontologies developed within WP7 for the fishery use case: *commodities_v1.0.owl*[14], *fi.owl*[15], *fishing_areas_v1.0.owl*[16], *gears_v1.0.owl*[17], *land_v1.0.owl*[18], *species_v1.0.owl*[19], and *vessels_v1.0.owl*[20].
- Three large ontologies: NCI Thesaurus[21] version 6.10d, Full-Galen[22] and CyC[23].

Any ontology to be used as input for the interoperability evaluation must be previously assessed in order to detect possible problems that could lead to mistakes or inconsistencies in the analysis of the interoperability results.
We performed a first analysis of the ontologies in three steps:

1. First, we checked the OWL species of the ontologies using the WonderWeb OWL Ontology Validator[24] and the Swoop[25] ontology editor (having Pellet as reasoner).
2. Second, we checked the consistency of the ontologies using Swoop and classifying the ontology with Pellet.
3. Third, we imported the ontologies into the NeOn Toolkit by hand.

Table 2.1 presents a summary of the results of this analysis, including the size in bytes of the ontologies.

- Regarding the OWL species of the ontologies, they range from OWL Lite to OWL Full and in two cases the ontologies could not be loaded in both the OWL Validator and Swoop.
- Regarding the consistency of the ontologies, only three of the ontologies are consistent and the rest either are inconsistent or cannot be classified into Swoop.
- Finally, most of the ontologies can be loaded into the NeOn Toolkit, the exception are four ontologies that produce parse errors when loading.

Clearly, we cannot evaluate interoperability with ontologies that cannot be loaded into tools or have parse problems. This leaves us with only six ontologies: *aos.owl*, *asc.owl*, *languagecode.owl*, *commodities_v1.0.owl*, *gears_v1.0.owl*, and *vessels_1.0.owl*.

[14] http://www.fao.org/aims/aos/fi/commodities_v1.0.owl
[15] http://www.fao.org/aims/aos/fi/fi.owl
[16] http://www.fao.org/aims/aos/fi/fishing_areas_v1.0.owl
[17] http://www.fao.org/aims/aos/fi/gears_v1.0.owl
[18] http://www.fao.org/aims/aos/fi/land_v1.0.owl
[19] http://www.fao.org/aims/aos/fi/species_v1.0.owl
[20] http://www.fao.org/aims/aos/fi/vessels_v1.0.owl
[21] ftp://ftp1.nci.nih.gov/pub/cacore/EVS/NCI_Thesaurus/archive/06.10d_Release/
[22] http://www.co-ode.org/galen/full-galen.owl
[23] http://www.cyc.com/2004/06/04/cyc
[24] http://www.mygrid.org.uk/OWL/Validator
[25] http://code.google.com/p/swoop/

Ontology	Size	OWL species	Consistency	Loads in NeOn Toolkit
AIMS				
aos.owl	73.4 Kb	DL	Consistent	OK
asc.owl	815 Kb	Full	Inconsistent	OK
languagecode.owl	6.36 Mb	DL	Consistent	OK
WP7				
commodities_v1.0.owl	10.3 Mb	DL	Consistent	OK
fi.owl	3.2 Mb	Does not load	Does not classify	Parse error
fishing_areas_v1.0.owl	127 Kb	Lite	Inconsistent	Parse error
gears_v1.0.owl	79.6 Kb	Lite	Inconsistent	OK
land_v1.0.owl	255 Kb	Full	Inconsistent	Parse error
species_v1.0.owl	13.1 Mb	DL	Does not classify	OK
vessels_v1.0.owl	103 Kb	DL	Inconsistent	OK
Large ontologies				
cyc.owl	23.3 Mb	Full	Does not classify	Parse error
full-galen.owl	20.1 Mb	Full	Does not classify	OK
NCIThesaurus.owl	75.8 Mb	Does not load	Does not classify	Parse error

Table 2.1: Analysis of the ontologies provided for the interoperability evaluation.

Chapter 3

RDF(S) interoperability evaluation

In this chapter we present the analysis of the interoperability of the NeOn Toolkit with other Semantic Web technologies using RDF(S) as interchange language. The tools evaluated and their corresponding versions can be seen in Table 3.1.

Tool	Version	Developer
Jena	2.3	HP
NeOn Toolkit	1.2.0 B739	NeOn Project
WebODE	2.0 build 140	Universidad Politécnica de Madrid

Table 3.1: Tools participating in the RDF(S) interoperability evaluation.

This analysis is performed in two consecutive steps:

1. We make a description of the behaviour of the NeOn Toolkit in the combined operation of importing one RDF(S) ontology and exporting it again (a step of the experiment as defined in Section 2.1).
2. With the information of the behaviour of the NeOn Toolkit in a step of the experiment, we also provide the analysis of the interoperability of the NeOn Toolkit with all the tools participating in the benchmarking (including itself).

Additionally, and within the analysis, we provide references to the ontology or ontologies that originated the comment with their names between parentheses, i.e., *(graph01-03)*.

3.1 RDF(S) compliance results

For analysing the behaviour of the NeOn Toolkit in one single step of the experiment (a combined import and export operation), we have considered the results of the NeOn Toolkit when it is the origin of the interchange (*Step 1*), irrespective of the tool that is the destination of the interchange. This step has as input one original ontology that is imported by the NeOn Toolkit and then exported into a resultant ontology.

The different step executions do not produce any exception in the NeOn Toolkit, and thus, a resultant ontology is always generated. Besides, the original and the resultant ontologies are always equivalent for all the combinations of components.

3.2 RDF(S) interoperability results

For analysing the interoperability of the NeOn Toolkit with the other tools participating in the benchmarking (including itself), we have considered the results of the interoperability of the NeOn Toolkit when it is the origin and the destination of the interchange with all the other tools.

Table 3.2[1] gives an overview of the interoperability between the tools and shows the percentage of benchmarks in which the original (O_i) and the resultant (O_i^{IV}) ontologies in an interchange are the same. For each cell, the row indicates the tool origin of the interchange, whereas the column indicates the tool destination of the interchange.

		DESTINATION		
		JE	**NT**	**WE**
ORIGIN	**JE**	100	100	30
	NT	100	100	30
	WE	30	30	30

Table 3.2: Percentage of identical interchanged ontologies in the RDF(S) evaluation.

Table 3.3 shows a summary of the results of the interoperability of the NeOn Toolkit with the other tools. In this table, the results of the interoperability between two tools (i.e., T1 and T2) have been grouped in categories and they include the interchange from one tool to another (from T1 to T2) and vice versa (from T2 to T1). The results in the table are restrictive, i.e., when a single benchmark in a category has a problem in one of the directions of the interchange, the whole category has this problem. The results for a category can be the following:

- **SAME.** When all the ontologies interchanged between two tools are the same (all the benchmarks in the category have an *INTEROPERABILITY* result of *SAME*).
- **DIFF.** When at least one ontology interchanged between two tools is different and execution errors do not exist (any benchmark in the category has an *INTEROPERABILITY* result of *DIFFERENT* and there is no benchmark with an *EXECUTION* result of *N.E.*).
- **N.E.** When at least one ontology could not be interchanged between two tools because of an execution error (any benchmark in the category has a *EXECUTION* result of *N.E.* - Non Executed).

In Table 3.3 we can see that the results of the interoperability of the **NeOn Toolkit with itself and with Jena** indicate that these tools can interchange all the combinations of components present in the benchmarks.

The results of the interoperability of the **NeOn Toolkit with WebODE** depend on the behaviour of WebODE. The combinations of components that can be successfully interchanged between the NeOn Toolkit and WebODE are exactly the same than those that WebODE can successfully interchange with itself. These combinations of components are the following:

- Classes.
- Class hierarchies without cycles.
- Properties with range *String* or an XML Schema datatype and a single domain.
- Instances of single classes.
- Instances with a property with a literal value with the property definition.

[1] Tool names have been shortened in the tables: JE=Jena, NT=NeOn Toolkit, and WE=WebODE.

Categories	**Benchmarks**	**JE-NT**	**NT-NT**	**WE-NT**
Classes	graph01-02	SAME	SAME	SAME
Classes instance of metaclasses	graph03-07	SAME	SAME	DIFF
Class hierarchies without cycles	graph08-10	SAME	SAME	SAME
Class hierarchies with cycles	graph11-12	SAME	SAME	DIFF
Classes related with classes through properties	graph13-16	SAME	SAME	DIFF
Classes related with literals through properties	graph17-18	SAME	SAME	DIFF
Properties and property hierarchies	graph19-25	SAME	SAME	DIFF
Properties with undefined domain or range	graph26	SAME	SAME	DIFF
Properties without domain or range	graph27-38	SAME	SAME	DIFF
Properties with single or multiple domains and ranges	graph39-43	SAME	SAME	DIFF
Properties range *String* or an XML Schema datatype and with a single domain	graph44,46	SAME	SAME	SAME
Properties with range *String* or an XML Schema datatype and with multiple domains	graph45,47	SAME	SAME	DIFF
Properties with domain *rdfs:Class*	graph48-49	SAME	SAME	DIFF
Instances of undefined resources	graph50	SAME	SAME	DIFF
Instances of single classes	graph51,53	SAME	SAME	SAME
Instances of multiple classes	graph52	SAME	SAME	DIFF
Instances with a property with a class value without the property definition	graph54-55	SAME	SAME	DIFF
Instances with a property with a literal value without the property definition	graph56	SAME	SAME	DIFF
Instances with a property with a class value with the property definition	graph57-63	SAME	SAME	DIFF
Instances with a property with a literal value with the property definition	graph64-67	SAME	SAME	SAME

Table 3.3: Summary of the results of the RDF(S) interoperability of the NeOn Toolkit.

3.3 Evolution of RDF(S) interoperability results

This section compares the results of the NeOn Toolkit in its last version (version 1.2.0 B739) to its previous results. We present first the RDF(S) compliance results and, second, the RDF(S) interoperability results with all the tools (including the NeOn Toolkit itself).

Regarding the RDF(S) compliance of the NeOn Toolkit, table 3.4 presents the results of a step execution for the NeOn Toolkit before and after the changes; it shows the number of benchmarks in each category in which the results of a step execution can be classified. In such results we can observe that the updated version of the NeOn Toolkit always produces the same ontologies in the first step of the experiment and that the problems previously detected have been removed.

	NeOn Toolkit v1.0 B823	**NeOn Toolkit v1.2.0 B739**
Same	42	82
More	16	
Less	21	
Tool fails	1	
Comparer fails		
Not valid	2	
TOTAL	82	82

Table 3.4: Updated results in *Step 1* in the RDF(S) evaluation.

As seen in the previous section, this improvement in the RDF(S) compliance has caused an improvement in the NeOn Toolkit interoperability. If we compare the percentage of benchmarks in which the original and the resultant ontologies in an interchange are the same (see table 3.5 for the previous results and table 3.2 for the current ones), we can see that the current version of the NeOn Toolkit is able of interchanging correctly all the ontologies with Jena and with itself. Besides, previously it could not interchange any combination of components with WebODE and now it can interchange some of them (see table 3.3).

		DESTINATION		
		JE	**NT**	**WE**
ORIGIN	**JE**	100	51	
	NT	51	51	
	WE			

Table 3.5: Percentage of identical interchanged ontologies in the previous RDF(S) evaluation.

Chapter 4

OWL interoperability evaluation

In this chapter we present the analysis of the interoperability of the NeOn Toolkit with other Semantic Web technologies using OWL as interchange language. The tools evaluated and their corresponding versions can be seen in Table 4.1.

Tool	Version	Developer
GATE	4.0	Sheffield University
Jena	2.3	HP
KAON2	2006-09-22	Karlsruhe University
NeOn Toolkit	1.2.0 build 739	The NeOn project
Protégé	3.3 build 395	Stanford University
Protégé-OWL	3.3 build 395	Manchester University
SemTalk	2.3	Semtation
SWI-Prolog	5.6.35	University of Amsterdam
WebODE	2.0 build 140	Universidad Politécnica de Madrid

Table 4.1: Tools participating in the OWL interoperability evaluation.

This analysis is carried out in two consecutive steps:

1. We describe the behaviour of the NeOn Toolkit in the combined operation of importing one OWL ontology and exporting it again (a step of the experiment as defined in Section 2.1).
2. With the information about the behaviour of the NeOn Toolkit during a step of the experiment, we provide the analysis of the interoperability of the NeOn Toolkit with all the tools participating in the benchmarking (including itself).

Additionally, within the analysis we provide references to the ontology or ontologies that originated the comment with their names between parentheses, i.e. *(ISA01-ISA03)*.

4.1 OWL compliance results

For analysing the behaviour of the NeOn Toolkit in one single step of the experiment (a combined import and export operation), we have considered the results of the NeOn Toolkit when it is the origin of the interchange (*Step 1*), irrespective of the tool that is the destination of the interchange. This step has as input one original ontology that is imported by the NeOn Toolkit and then exported into a resultant ontology.

The different step executions do not produce any exception in the NeOn Toolkit, and thus, a resultant ontology is always generated. The results of a step execution in the NeOn Toolkit can be classified in three types:

- *The resultant ontology includes more information than the original one.* This happens in the 11 cases where the original ontology contains properties with a range of *rdfs:Literal* (ISE08-10, ISF02, ISG04, ISI04-05, ISL07-08, ISL11-14).

  In these cases, the NeOn Toolkit inserts the triple "*rdfs:Literal rdf:type owl:Datatype.*" in the ontology, but *owl:Datatype* is not present in the OWL vocabulary.

- *The resultant ontology includes less information than the original one.* In this case, information is also inserted in the former ontology. This occurs in the 5 cases where blank nodes appear in the ontology (ISJ01-03, ISL13-14).

  In these cases, the NeOn Toolkit converts the blank nodes into named nodes (e.g., converts "*_:7beb rdf:type ns:Person.*" into "*ns:genid1 rdf:type ns:Person.*".

- *The original and the resultant ontologies are equivalent.* This happens in the rest of the cases.

In summary, we can say that the NeOn Toolkit is compliant with OWL Lite with the exception of ontologies that include blank nodes and ontologies that include properties with a range of *rdfs:Literal*. Nevertheless, we must take into account that the OWL Lite Import Benchmark Suite does not exhaustively cover the OWL Lite knowledge model and that other problems can be detected with new combinations of components not included in the benchmark suite.

4.2 OWL interoperability results

For analysing the interoperability of the NeOn Toolkit with the other tools participating in the benchmarking (including itself), we have considered the results of the interoperability of the NeOn Toolkit when it is the origin and the destination of the interchange with all the other tools.

Table 4.2[1] gives an overview of the interoperability between the tools and shows the percentage of benchmarks in which the original (O_i) and the resultant (O_i^{IV}) ontologies in an interchange are the same. For each cell, the row indicates the tool origin of the interchange, whereas the column indicates the tool destination of the interchange.

		DESTINATION								
		JE	**PO**	**SP**	**NT**	**KA**	**GA**	**ST**	**WE**	**PF**
ORIGIN	**JE**	100	100	100	80	58	70	0	31	4
	PO	100	100	95	80	58	78	0	31	4
	SP	100	100	100	80	58	91	46	31	4
	NT	80	80	80	80	58	84	57	31	4
	KA	58	58	58	58	58	67	45	19	13
	GA	92	92	75	78	56	60	0	29	25
	ST	41	41	46	48	36	39	40	34	0
	WE	31	31	0	31	19	29	20	31	20
	PF	4	4	0	4	0	3	0	4	4

Table 4.2: Percentage of identical interchanged ontologies in the OWL evaluation.

Table 4.3 shows the results from the interoperability of the NeOn Toolkit with the other tools. In this table, the results of the interoperability between two tools (i.e., T1 and T2) have been grouped in categories, as in the previous section, and they include the interchange from one tool to another (from T1 to T2) and vice versa (from T2 to T1). The results in the table are restrictive, i.e., when a single benchmark in a category has a problem in one of the directions of the interchange, then the whole category has this problem.

[1]Tool names have been shortened in the tables: GA=GATE, JE=Jena, K2=KAON2, NT=NeOn Toolkit, PF=Protégé Frames, PO=Protégé OWL, ST=SemTalk, SP=SWI-Prolog, and WE=WebODE.

Categories	Benchmarks	GA-NT	JE-NT	K2-NT	NT-NT	PF-NT	PO-NT	ST-NT	SP-NT	WE-NT
Class hierarchies										
Named class hierarchies without cycles	ISA01-ISA04	SAME	SAME	DIFF	SAME	DIFF	SAME	SAME	SAME	SAME
Named class hierarchies with cycles	ISA05-ISA06	SAME	SAME	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF
Classes subclass of a value constraint in an object property	ISA07-ISA08	SAME	SAME	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF
Classes subclass of a cardinality constraint in an object property	ISA09-ISA12	SAME	SAME	N.E.	SAME	DIFF	SAME	N.E.	SAME	DIFF
Classes subclass of a cardinality constraint in a datatype property	ISA13-ISA16	DIFF	SAME	N.E.	SAME	N.E.	SAME	DIFF	SAME	DIFF
Classes subclass of a class intersection	ISA17	SAME	SAME	SAME	SAME	DIFF	SAME	SAME	SAME	DIFF
Class equivalences										
Equivalent named classes	ISB01	SAME	SAME	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF
Classes equivalent to a value constraint in an object property	ISB02-ISB03	SAME	SAME	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF
Classes equivalent to a cardinality constraint in an object property	ISB04-ISB07	SAME	SAME	N.E.	SAME	DIFF	SAME	N.E.	SAME	DIFF
Classes equivalent to a cardinality constraint in a datatype property	ISB08-ISB11	DIFF	SAME	N.E.	SAME	N.E.	SAME	DIFF	SAME	DIFF
Classes equivalent to a class intersection	ISB12	SAME	SAME	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF
Classes defined with set operators										
Classes intersection of other classes	ISC01-ISC02	SAME	SAME	SAME	SAME	DIFF	SAME	SAME	SAME	DIFF
Property hierarchies										
Object property hierarchies	ISD01-ISD02	SAME	SAME	DIFF	SAME	DIFF	SAME	SAME	SAME	DIFF
Datatype property hierarchies	ISD03-ISD04	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF	SAME	DIFF
Properties with domain and range										
Object properties without domain or range	ISE01-ISE02	SAME	SAME	SAME	SAME	DIFF	SAME	SAME	SAME	SAME
Object properties with domain and range	ISE03-ISE04	SAME	SAME	SAME	SAME	DIFF	SAME	SAME	SAME	SAME
Object properties with multiple domains or ranges	ISE05-ISE06	SAME	SAME	SAME	SAME	DIFF	SAME	SAME	SAME	DIFF
Datatype properties without domain or range	ISE07-ISE08	DIFF	DIFF	DIFF	DIFF	N.E.	DIFF	DIFF	DIFF	DIFF
Datatype properties with domain and range	ISE09	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF	SAME	DIFF	DIFF
Datatype properties with multiple domains	ISE10	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF	SAME	DIFF	DIFF
Relations between properties										
Equivalent object and datatype properties	ISF01-ISF02	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF	SAME	DIFF	DIFF
Inverse object properties	ISF03	SAME	SAME	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF
Global cardinality constraints and logical property characteristics										
Transitive object properties	ISG01	SAME	SAME	SAME	SAME	DIFF	SAME	SAME	SAME	SAME
Symmetric object properties	ISG02	SAME	SAME	SAME	SAME	DIFF	SAME	SAME	SAME	SAME
Functional object and datatype properties	ISG03-ISG04	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF
Inverse functional object properties	ISG05	SAME	SAME	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF
Single individuals										
Instances	ISH01, ISH03	DIFF	SAME	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF
Instances of multiple classes	ISH02	DIFF	SAME	SAME	SAME	DIFF	SAME	SAME	SAME	SAME
Named individuals and properties										
Named individuals and object properties	ISI01-ISI03	SAME	SAME	SAME	SAME	DIFF	SAME	SAME	SAME	SAME
Named individuals and datatype properties	ISI04-ISI05	DIFF	DIFF	DIFF	DIFF	N.E.	DIFF	DIFF	DIFF	DIFF
Anonymous individuals and properties										
Anonymous individuals and object properties	ISJ01-ISJ02	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF	DIFF
Anonymous individuals and datatype properties	ISJ03	DIFF	DIFF	DIFF	DIFF	N.E.	DIFF	DIFF	DIFF	DIFF
Individual identity										
Equivalent individuals	ISK01	SAME	SAME	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF
Different individuals	ISK02-ISK03	SAME	SAME	SAME	SAME	DIFF	SAME	DIFF	SAME	DIFF

Table 4.3: Summary of the results of the OWL interoperability of the NeOn Toolkit.

The results for a category can be the following:

- **SAME.** When all the ontologies interchanged between two tools are the same (all the benchmarks in the category have an *INTEROPERABILITY* result of *SAME*).
- **DIFF.** When at least one ontology interchanged between two tools is different and there were no execution errors (any benchmark in the category has an *INTEROPERABILITY* result of *DIFFERENT* and no benchmark exists with an *EXECUTION* result of *N.E.*).
- **N.E.** When at least one ontology could not be interchanged between two tools because of an execution error (any benchmark in the category has an *EXECUTION* result of *N.E.* - Non Executed).

In Table 4.3 we can see that the interoperability results of the **NeOn Toolkit with itself, Jena, Protégé OWL and SWI-Prolog** depend on the behaviour of the NeOn Toolkit described in the previous section. Therefore, the NeOn Toolkit can interchange with these tools all the combinations of components of the OWL Lite knowledge model taken into account in the evaluation with the exception of ontologies that include blank nodes and ontologies that include properties with a range of *rdfs:Literal*.

The interoperability results of the **NeOn Toolkit with GATE** depend on the behaviour of the NeOn Toolkit but also on the behaviour of GATE. This makes that the NeOn Toolkit can interchange with GATE all the combinations of components of the OWL Lite knowledge model taken into account in the evaluation with the exception of ontologies that include: blank nodes, properties with a range of *rdfs:Literal*, cardinality constraints in datatype properties, or instances of single or multiple classes.

The interoperability results of the **NeOn Toolkit with KAON2** depend on the behaviour of the NeOn Toolkit but also on the behaviour of KAON2. This makes that the NeOn Toolkit can interchange with KAON2 all the combinations of components of the OWL Lite knowledge model taken into account in the evaluation with the exception of ontologies that include: blank nodes, properties with a range of *rdfs:Literal*, named class hierarchies without cycles, cardinality constraints, or property hierarchies.

The interoperability results of the **NeOn Toolkit with Protégé Frames** depend on the behaviour of the NeOn Toolkit but also on the behaviour of Protégé Frames. Therefore, the NeOn Toolkit cannot interchange any combination of components with Protégé Frames.

The interoperability results of the **NeOn Toolkit with SemTalk** depend on the behaviour of the NeOn Toolkit but also on the behaviour of SemTalk. Therefore, the only combinations of components that the NeOn Toolkit can interchange with SemTalk are: named class hierarchies with cycles, classes subclass of a class intersection, classes intersection of other classes, object property hierarchies, object properties with or without domain or range and with multiple domains or ranges, datatype properties with domain or range and with multiple domains, equivalent object and datatype properties, transitive and symmetric object properties, instances of multiple classes, and named individuals and object properties.

The interoperability results of the **NeOn Toolkit with WebODE** depend on the behaviour of the NeOn Toolkit but also on the behaviour of WebODE. Therefore, the only combinations of components that the NeOn Toolkit can interchange with WebODE are: named class hierarchies without cycles, object properties with or without domain or range, transitive and symmetric object properties, instances of multiple classes, and named individuals and object properties.

4.3 Evolution of OWL interoperability results

This section compares the results of the NeOn Toolkit in its last version (version 1.2.0 B739) to its previous results. We present first the evolution of the OWL compliance results and, second, the evolution of the OWL interoperability results with all the tools (including the NeOn Toolkit itself).

Since we have changed the comparer used for analysing the results, current results are not comparable to previous ones. Therefore, we executed again the experiments with the previous version of the NeOn Toolkit in order to correctly compare the results[2]. Nevertheless, it must be noted that results are comparable only to a certain extent because of the significant changes performed inside the NeOn Toolkit from the previous version to the current one.

Regarding the OWL compliance of the NeOn Toolkit, table 4.4 presents the results of a step execution for the NeOn Toolkit before and after the changes; it shows the number of benchmarks in each category in which the results of a step execution can be classified. In such results we can observe that the updated version of the NeOn Toolkit performs better than the previous one, producing in most of the cases the same ontologies in the first step of the experiment and removing the problems previously detected.

	NeOn Toolkit v1.0 B823	**NeOn Toolkit v1.2.0 B739**
Same	23	66
More		11
Less	59	5
Tool fails		
Comparer fails		
Not valid		
TOTAL	82	82

Table 4.4: Updated results in *Step 1* in the OWL evaluation.

As seen in the previous section, this improvement in the OWL compliance has caused an improvement in the NeOn Toolkit interoperability.

If we compare the percentage of benchmarks in which the original and the resultant ontologies in an interchange are the same (see table 4.5 for the previous results and table 4.2 for the current ones), we can see that interoperability has notably improved. For example, the percentage of equivalent interchanged ontologies with the OWL-native tools has increased from 28 to 80 percent. Besides, previously in some cases no interchanges were possible with some tools (i.e., Protégé Frames and SemTalk) but now the NeOn Toolkit can interchange some ontologies with them.

		DESTINATION								
		JE	**PO**	**SP**	**NT**	**KA**	**GA**	**ST**	**WE**	**PF**
ORIGIN	**JE**	100	100	100	28	58	70	0	31	4
	PO	100	100	95	28	58	78	0	31	4
	SP	100	100	100	28	58	91	46	31	4
	NT	28	28	28	28	15	28	15	15	0
	KA	58	58	58	15	58	67	45	19	13
	GA	92	92	75	23	56	60	0	29	25
	ST	41	41	46	0	36	39	40	34	0
	WE	31	31	0	21	19	29	20	31	20
	PF	4	4	0	4	0	3	0	4	4

Table 4.5: Percentage of identical interchanged ontologies in the previous OWL evaluation.

4.4 OWL interoperability results with real-world ontologies

In section 2.3 we saw that we could only use a few of the real-world ontologies provided for extending the interoperability evaluation. We executed the IBSE tool with all the ontologies to obtain as much results as possible about the OWL compliance of the NeOn Toolkit but, as can be seen in table 4.6, the execution

[2]This causes that the figures presented here do not correspond to those in the previous deliverable.

of the first step in the IBSE tool was only correct for a few cases: the three consistent ontologies and two inconsistent ones.

Ontology	**Execution in IBSE**
AIMS	
aos.owl	OK
asc.owl	Fail
languagecode.owl	OK
WP7	
commodities_v1.0.owl	OK
fi.owl	Fail
fishing_areas_v1.0.owl	Fail
gears_v1.0.owl	OK
land_v1.0.owl	Fail
species_v1.0.owl	Fail
vessels_v1.0.owl	OK
Large ontologies	
cyc.owl	Fail
full-galen.owl	Fail
NCIThesaurus.owl	Fail

Table 4.6: Results of executing IBSE with the extended ontologies.

The results of a step execution in the NeOn Toolkit with these ontologies can be classified into two types:

- *The resultant ontology includes more information than the original one*. This happens in the 5 cases where the original ontology contains properties with a XML Schema datatype as range: *xsd:date*, *xsd:int*, *xsd:boolean*, *xsd:string*, *xsd:dateTime*, etc. (*aos.owl*, *commodities_v1.0.owl*, *languagecode.owl*, *gears_v1.0.owl*, and *vessels_v1.0.owl*).

  In these cases, the NeOn Toolkit inserts the triple "*xsd:<datatype> rdf:type owl:Datatype.*" in the ontology, but *owl:Datatype* is not present in the OWL vocabulary.

  Besides, when the NeOn Toolkit finds an enumerated datatype[3] whose data values are not defined as a *rdf:List* (*aos.owl*), it defines the data values as a list including the triple *"<node> rdf:type rdf:List"*.

- *The resultant ontology includes less information than the original one*. In this case, information is also inserted in the former ontology. This occurs in 1 case where the minimal cardinality of a restriction is defined with a datatype of *xsd:int* (*languagecode.owl*).

  In this case, the NeOn Toolkit converts the datatype *xsd:int* into the datatype *xsd:nonNegativeInteger*.

Since in none of the cases the NeOn Toolkit produced equivalent ontologies and some of the ontologies are inconsistent, we only evaluated the OWL compliance of the NeOn Toolkit and not its interoperability.

[3] http://www.w3.org/TR/2004/REC-owl-ref-20040210/#EnumeratedDatatype

Chapter 5

Recommendations for improving the NeOn Toolkit

The following recommendations are intended to improve the interoperability of the NeOn Toolkit using OWL as the interchange language. These recommendations have been extracted from the analysis presented in Chapter 4. We do not provide any recommendation to improve the interoperability using RDF(S) as the interchange language because the results in Chapter 3 show no problems.

Though it is not compulsory to follow these recommendations, they would improve the interoperability of the NeOn Toolkit in the identified situations; it has to be noted that, in some cases, the results present the intended behaviour of the tool as programmed by its developers and the tool is working correctly.

In order to increase its interoperability using OWL as interchange language, the NeOn Toolkit should do the following:

- Should not export the triple "*<datatype> rdf:type owl:Datatype.*" when an ontology contains properties with a range of *rdfs:Literal* or a XML Schema Datatype because *owl:Datatype* is not present in the OWL vocabulary.
- Should avoid or minimise the possibility of having interoperability problems because of naming blank nodes.

  The NeOn Toolkit names blank nodes with the name *genid<num>*, where *<num>* are integers starting from 1. If two ontologies with the same namespace and with blank nodes are imported, each of them would have at least the *ns:genid1* node and this would cause to consider the two blank nodes being the same.

  Even if the NeOn Toolkit cannot represent internally blank nodes, several solutions to this problem could be suggested:

  - To internally "mark" certain nodes as blank nodes so they can be exported again as blank nodes.
  - To generate random integers of a certain size for naming blank nodes inside the NeOn Toolkit instead of using integers starting from 1. This would not completely remove the interoperability problem mentioned before but it would be quite unlikely to happen.
- Should export values of cardinality restrictions with a datatype of *xsd:integer*. There is no problem with the current behaviour of the NeOn Toolkit when it exports values of cardinality restrictions with a datatype of *xsd:nonNegativeInteger*. Nevertheless, the OWL recommendation states that *xsd:string* and *xsd:integer* are the minimal XML Schema datatypes that must be supported by tools[1] and, therefore, using these datatypes instead of similar ones should improve the interoperability with other tools.

[1] `http://www.w3.org/TR/owl-ref/#DatatypeSupport`

Chapter 6

Conclusions

This deliverable presents the results of the second iteration of the evaluation of the interoperability of the NeOn Toolkit with other Semantic Web tools in two different scenarios, having in each RDF(S) and OWL as interchange language, respectively.

The results show that the interoperability of the NeOn Toolkit has significantly improved since it has native OWL support. Nevertheless, some issues still prevent the NeOn Toolkit of being a full OWL-interoperable tool. Although these issues do not prevent a correct working of the NeOn Toolkit, they hinder its interoperability with other tools and its adoption by a broad range of users.

These results have provided a set of recommendations to improve the interoperability of the NeOn Toolkit. However, it has to be taken into account that the interoperability problems encountered were not only caused by the NeOn Toolkit, as it has been observed that in many cases interoperability problems were created by other tools.

We have also presented the improvement in time of the OWL interoperability results of the NeOn Toolkit, showing how the improvement of the NeOn Toolkit also entails the improvement of the interoperability of this tool with the others. Nevertheless, it must be noted that this requires to consider the benchmarking of the interoperability of the NeOn Toolkit as a continuous activity.

Including new real-world ontologies has provided us with new insights of the behaviour and interoperability of the NeOn Toolkit. Besides, it has highlighted the need of further extending the interoperability evaluation, either including new benchmarks in the existing benchmark suites (e.g., considering XML Schema datatypes) or developing new benchmark suites (e.g., considering OWL DL, OWL Full, OWL 2, etc.).

These extensions of the benchmark suites will probably provoke changes in the evaluation infrastructure. For example, OWL Full is not managed by current OWL reasoners, which are DL-based. Therefore, a comparison at the conceptual level is not possible with current technologies. Some solutions to this problem could be, for example, to compare ontologies at the level of RDF triples or to use ontologies with simple combinations of OWL Full components.

We have also observed that current tools are not able of managing large ontologies. Nevertheless, the IBSE tool has proved useful for comparing large ontologies. For example, when comparing the *languagecode.owl* ontology (with a size of 6.36 Mb) with the exported ontology (with a size of 4.5 Mb) it identified the 6 differing triples and allowed an instantaneous analysis of the behaviour of the tool.

Finally, we have learned that the indiscriminate use of real-world ontologies is not advisable for evaluating interoperability; only 6 of the 13 proposed ontologies were expected to provide reliable results because they could be loaded into tools and posed no parse problems. Nevertheless, using real-world ontologies has proved useful to detect new cases not covered by current benchmarks.

Bibliography

[Bar07] J. Barrasa. *Modelo para la definición automática de correspondencias semánticas entre ontologías y modelos relacionales*. PhD thesis, Universidad Politécnica de Madrid. Facultad de Informática, January 2007.

[GC07] R. García-Castro. D6.8.1: Testing the NeOn Toolkit interoperability. Technical report, NeOn, September 2007.

[GCGP08] R. García-Castro and A. Gómez-Pérez. Large-scale benchmarking of the OWL interoperability of Semantic Web technologies. In *Proceedings of the Second IEEE International Conference on Semantic Computing (ICSC2008)*, Santa Clara, CA, USA, August 4-7 2008. IEEE Computer Society.

[GCGPS07] R. García-Castro, A. Gómez-Pérez, and Y. Sure. Benchmarking the RDF(S) interoperability of ontology tools. In *Proceedings of the Nineteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2007)*, pages 410–415, Boston, Massachusetts, USA, July 9-11 2007. Knowledge Systems Institute Graduate School.

[GCMW+04] R. García-Castro, D. Maynard, H. Wache, D. Foxvog, and R. González-Cabero. D2.1.4 specification of a methodology, general criteria and benchmark suites for benchmarking ontology tools. Technical report, Knowledge Web, December 2004.