Perspectives in Semantic Interoperability

Raúl García-Castro[1] and Asunción Gómez-Pérez[2]

Ontology Engineering Group
[1]Departamento de Lenguajes y Sistemas Informáticos e Ingeniería Software
[2]Departamento de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain
{rgarcia, asun}@fi.upm.es

Abstract. This paper describes the problem of semantic technology interoperability from two different perspectives. First, from a theoretical perspective by presenting an overview of the different factors that affect interoperability and, second, from a practical perspective by reusing evaluation methods and applying them to six current semantic technologies in order to assess their interoperability.

1 Introduction

Due to the high heterogeneity in semantic technologies, achieving interoperability between the growing number of semantic technologies is not straightforward. Furthermore, the real interoperability capabilities of the current semantic technologies are unknown and this hinders the development of semantic systems, either when designing their interactions with other systems or when selecting the right semantic components to be used inside a system.

This converts interoperability assurance and evaluation into two key needs when developing semantic systems. These needs are currently unfulfilled mainly due to the lack of updated information on the interoperability of existing semantic technologies.

The goal of this paper is to describe in detail the problem of interoperability between semantic technologies from a theoretical perspective, by presenting an overview of the different factors that affect interoperability, and also from a practical one, by using existing evaluation methods and applying them to six current semantic technologies to assess their interoperability.

One of the pillars to base interoperability upon is standards, such as the RDF(S) and OWL ontology language specifications defined in the W3C. Hence, the conformance of semantic technologies to these standards is a main characteristic to evaluate when gathering information about their interoperability.

This paper is structured as follows. Section 2 provides an overview of the problem of interoperability between semantic technologies. Section 4 describes two approaches used to evaluate the conformance and interoperability of six different semantic technologies; the results of these evaluations are presented in sections 5 and 6, respectively. Finally, section 7 draws some conclusions from the work presented in this paper.

2 Semantic Technology Interoperability

According to the IEEE, interoperability is the ability of two or more systems or components to exchange information and to use this information [1]. Duval proposes a similar definition by stating that interoperability is the ability of independently developed software components to exchange information so they can be used together [2]. For us, interoperability is the ability that semantic systems have to interchange ontologies and use them.

One of the factors that affects interoperability is heterogeneity. Sheth [3] classifies the levels of heterogeneity of any information system into information heterogeneity and system heterogeneity; in this paper, only information heterogeneity (and, therefore, interoperability) is considered. Furthermore, interoperability is treated in this paper in terms of knowledge reuse and must not be confused with the interoperability problem caused by the integration of resources, the latter being related to the ontology alignment problem [4].

The main ontology management functionalities of semantic systems that are related to interoperability are the following:

- **Storing Ontologies.** Semantic systems need to store the ontologies they use, either if these ontologies define their main data model or if they are used as information objects.
- **Operating over Ontologies.** These stored ontologies need to be further processed (e.g., modified, visualised) to fulfil the system functionalities.
- **Importing Ontologies.** Semantic systems need mechanisms to import ontologies coming from external systems.
- **Exporting =ntologies.** Related to the previous item, semantic systems also require a means to export ontologies to other external systems.

These functionalities are usually implemented by different components in the system. Besides, when developing semantic systems it is frequent to reuse components that already implement one or several of these functionalities. For example, to import and export ontologies using some ontology management framework (e.g., Jena) and even to reuse this framework for storing the ontologies.

We can identify two types of interoperability depending on whether interoperability is required inside the semantic system limits or outside it.

- **Internal Interoperability.** As mentioned above, semantic systems are frequently developed by reusing components that provide semantic capabilities and, hence, all the components inside a semantic system should interoperate correctly. Internal interoperability is achieved by means of specific software developments that guarantee interoperability and should be ensured during the development of the system.
- **External Interoperability.** Semantic systems have to interact with other semantic systems to perform complex tasks, that is, semantic systems have to interoperate with external systems. External interoperability is achieved by interchanging ontologies by means of a shared resource and requires mechanisms to assess up to what extent this interoperability can be accomplished.

3 Aspects of Interoperability

Semantic technology interoperability is highly affected by the heterogeneity of the knowledge representation formalisms of the different existing systems, since each formalism provides different knowledge representation expressiveness and different reasoning capabilities, as it occurs in knowledge-based systems [5].

This heterogeneity can be seen not only in W3C ontology language specifications where we have different ontology languages: RDF(S), the OWL sublanguages (Lite, DL and Full) and the OWL 2 profiles (EL, QL and RL), but also in other models used to represent ontologies, such as the Unified Modeling Language[1] (UML), the Ontology Definition Metamodel[2] (ODM), or the Open Biomedical Ontologies[3] (OBO) language. Besides, this heterogeneity does not only appear between different systems; it may happen that different components inside a system have different knowledge representation formalisms.

As commented above, we need to interchange ontologies, either between semantic systems or between the components of a semantic system. Two factors influence this interchange: the knowledge representation language used by the systems/components and the way of serializing ontologies during the interchange.

However, the influence of the serialization is not a big issue since, even if we can find different serializations to be used with an ontology language, it is straightforward to find a common serialization to interchange ontologies; what makes ontology interchanges problematic is the heterogeneity in knowledge representation formalisms previously mentioned.

The two common ways of interchanging ontologies within semantic systems are either directly by storing the ontology in the destination system or indirectly by storing the ontology in a shared resource, such as a fileserver, a web server, or an ontology repository.

Most semantic systems natively manage a W3C recommended language, either RDF(S) or OWL, and sometimes both; however, some systems manage other representation formalisms. If the systems participating in an interchange (or the shared resource) have different representation formalisms, the interchange requires at least a translation from one formalism to the other. These ontology translations from one formalism to another formalism with different expressiveness cause information additions or losses in the ontology, once in the case of a direct interchange and twice in the case of an indirect one.

4 Evaluating Semantic Technology Interoperability

The most common way for the current semantic technologies to interoperate is by means of a shared resource where the ontology is stored in a certain ontology language. In order to evaluate interoperability using an interchange language, one characteristic to consider is the conformance of the tools when dealing with ontologies defined in that

[1] http://www.uml.org/

[2] http://www.omg.org/ontology/

[3] http://obofoundry.org/

language.

The next sections describe two approaches to evaluate semantic technology conformance and interoperability. Due to the lack of space we do not provide detailed descriptions of these approaches. These, however, can be found in [6].

4.1 Evaluating Conformance

The conformance evaluation has the goal of evaluating the conformance of semantic technologies with regards to ontology representation languages, that is, to evaluate up to what extent semantic technologies adhere to the specification of ontology representation languages.

During the evaluation, a common group of tests is executed in two steps. Starting with a file containing an ontology, the execution consists in importing the file with the ontology into the origin tool and then exporting the ontology to another file.

After a test execution, we have two ontologies in the ontology representation language, namely, the original ontology and the final ontology exported by the tool. By comparing these ontologies we can know up to what extent the tool conforms to the ontology language. From the evaluation results, the following three metrics for a test execution can be defined: *Execution*, which informs of the correct test execution; *Information added or lost*, which shows the information added to or lost from the ontology; and *Conformance*, which explains whether the ontology has been processed correctly with no addition or loss of information.

4.2 Evaluating Interoperability

The interoperability evaluation has the goal of evaluating the interoperability of semantic technologies in terms of the ability that such technologies have to interchange ontologies and use them.

In concrete terms, the evaluation takes into account the case of interoperability using an interchange language, that is, when an ontology is interchanged by storing it in a shared resource (e.g., a fileserver, a web server, or an ontology repository) and is formalised using a certain ontology language.

During the experiment, a common group of tests is executed in two sequential steps. Let start with a file containing an ontology. The first step consists in importing the file with the ontology into the origin tool and then exporting the ontology to a file. The second step consists in importing the file with the ontology exported by the origin tool into the destination tool and then exporting the ontology to another file.

After a test execution, we have three ontologies in the ontology representation language, namely, the original ontology, the intermediate ontology exported by the first tool, and the final ontology exported by the second tool. By comparing these ontologies we can know up to what extent the tools are interoperable. For each of the two steps and for the whole interchange we have metrics similar to those presented above for evaluating conformance. Therefore, we can use the *Execution* and *Information added and lost* metrics as well as an *Interoperability* one, which explains whether the ontology has been interchanged correctly with no addition or loss of information.

4.3 Running the Evaluations

In this paper, we aim to evaluate conformance and interoperability using as interchange languages RDF(S), OWL Lite and OWL DL. To this end, we will use three different test suites that contain synthetic ontologies with simple combinations of knowledge model components from these languages.

The RDF(S) and OWL Lite Import Test Suites are described in [7] and the OWL DL Import Test Suite is described in [8]. These test suites have been defined similarly in a manual way; the main difference between them is that the OWL DL test suite has been generated following a keyword-driven process that allows obtaining a more exhaustive test suite (with 561 tests compared to the 82 tests of the other test suites).

The evaluations described above are part of the evaluation services provided by the SEALS Platform[4], a research infrastructure that offers computational and data resources for the evaluation of semantic technologies; the mentioned test suites are also included in that platform. Once a tool is connected to the SEALS Platform, the platform can automatically execute the conformance and interoperability evaluations. We connected six well-known tools to the platform and by means of the SEALS Platform executed the required conformance (for every tool and using every test suite) and interoperability (for every tool with all the other tools and using every test suite) evaluations.

The six tools evaluated were three ontology management frameworks: Jena (version 2.6.3), the OWL API (version 3.1.0 1592), and Sesame (version 2.3.1); and three ontology editors: the NeOn Toolkit (version 2.3.2 using the OWL API version 3.0.0 1310), Protégé OWL (version 3.4.4 build 579), and Protégé version 4 (version 4.1 beta 209 using the OWL API version 3.1.0 1602). As can be seen, sometimes tools use ontology management frameworks for processing ontologies.

5 Conformance Results

This section presents the conformance results for the six tools evaluated. Table 1 presents the tool conformance results for RDF(S), OWL Lite and OWL DL, respectively[5]. The tables show the number of tests in each category in which the results of a test can be classified, depending on whether the original and the resultant ontologies are the same (*SAME*), are different (*DIFF*), or the tool execution fails (*FAIL*).

As can be observed in these tables, Jena and Sesame present no problems when processing the ontologies included in the test suites for the different languages. Therefore, no further comments will be made on these tools.

Besides, as previously mentioned, the NeOn Toolkit and Protégé 4 use the OWL API for ontology management.

The version of Protégé 4 evaluated uses a version of the OWL API that is almost contemporary to the one we evaluated. Hence, after analysing the results of Protégé 4 we

[4] http://www.seals-project.eu/seals-platform

[5] The tool names have been abbreviated in the tables: JE=Jena, NT=NeOn Toolkit, OA=OWL API, P4=Protégé 4, PO=Protégé OWL, and SE=Sesame.

[6] Not counting additions of *owl:Ontology*.

[7] Not counting additions of *owl:NamedIndividual*.

Table 1. Conformance results.

(a) RDF(S).

Category	JE	NT	OA	P4	PO[6]	SE
SAME	82	0	0	0	68	82
DIFF	0	82	82	82	14	0
FAIL	0	0	0	0	0	0
TOTAL	82	82	82	82	82	82

(b) OWL Lite.

Category	JE	NT[7]	OA[7]	P4[7]	PO	SE
SAME	82	78	80	80	73	82
DIFF	0	2	2	2	9	0
FAIL	0	2	0	0	0	0
TOTAL	82	82	82	82	82	82

(c) OWL DL.

Category	JE	NT[7]	OA[7]	P4[7]	PO	SE
SAME	561	549	549	549	429	561
DIFF	0	8	11	11	132	0
FAIL	0	4	1	1	0	0
TOTAL	561	561	561	561	561	561

reached the same conclusions that those obtained for the OWL API and the comments made for the OWL API are also valid for Protégé 4.

However, the version of the NeOn Toolkit evaluated uses a version of the OWL API that differs in some months to the one we evaluated. In general, from the results of the NeOn Toolkit we reached the same conclusions that those obtained from the OWL API. In the next sections we will only comment on those cases where the behaviour of the NeOn Toolkit and the OWL API differ.

5.1 RDF(S) Conformance

When the **OWL API** processes RDF(S) ontologies, it always produces different ontologies because it converts the ontologies into OWL 2. The changes performed over the ontologies are the following:

- Classes are transformed into OWL classes.
- Individuals are transformed into OWL 2 named individuals.
- Properties are transformed according to their use. If a property either relates two classes or two individuals or has as domain a class and as range another class (even if the range class is *rdfs:Literal* or an XML Schema Datatype), it is transformed into an OWL object property. If a property either relates one individual with a literal value or does not have domain and has a range of *rdfs:Literal* or an XML Schema datatype, it is transformed into an OWL datatype property. If conditions from these two groups appear, the property is created both as an object and a datatype property.
- Classes and properties that are no further described (i.e., the only statement made about a resource is that it is a class or a property) are lost.
- Undefined resources that either appear as domain or range of a property or that have an instance are defined as OWL classes.
- Classes related by a property or classes related to a literal value using a property are transformed into OWL 2 named individuals.
- A particular case of the previous case is that of metaclasses (i.e., when the property that relates the two classes is the *rdf:type* property). In this case, classes without

instances and that are instance of another class are transformed into OWL 2 named individuals. Besides, classes that have instances and that are instance of another class are defined both as OWL classes and as OWL 2 named individuals.

When **Protégé OWL** processes an RDF(S) ontology, the ontology is always created as an OWL ontology with a randomly generated name[8]. Regardless of this, different ontologies are produced when the ontology contains

- A property with an undefined resource as range. The undefined resource is created as a class.
- A literal value. The literal value is created with a datatype of *xsd:string* and, therefore, it is a different literal. According to the RDF specification, one requirement for literals to be equal is that either both or neither have datatype URIs[9].

5.2 OWL Lite Conformance

When the **OWL API** processes OWL Lite ontologies, it converts the ontologies into OWL 2. Since OWL 2 covers the OWL Lite specification, most of the times the OWL API produces the same ontologies. However, one effect of this conversion is that individuals are converted into OWL 2 named individuals.

The cases in which the ontologies are different occur when the ontology contains a named individual related through an object property to an anonymous individual, and this anonymous individual is related through a datatype property to a literal value. In this case, the named individual is related through the object property to an anonymous resource, another anonymous resource is related through a datatype property to a literal value, and the anonymous individual is not related to anything.

After analysing the results of the **NeOn Toolkit** we obtained the same conclusions that were previously presented for the OWL API with one exception. When the ontology contains an anonymous individual related to a named individual through an object property, the execution of the NeOn Toolkit fails.

When **Protégé OWL** processes an OWL Lite ontology, most of the times it produces the same ontology. The only exception is when the ontology contains a literal value. The literal value is created with a datatype of *xsd:string* and, therefore, it is a different literal. As mentioned above, one requirement for literals to be equal is that either both or neither have datatype URIs.

5.3 OWL DL Conformance

When the **OWL API** processes OWL DL ontologies, it converts the ontologies into OWL 2. Since OWL 2 covers the OWL DL specification, most of the times the OWL API produces the same ontologies. However, one effect of this conversion is that individuals are converted into OWL 2 named individuals.

The cases when the ontologies are different occur when the ontology contains

[8] E.g., http://www.owl-ontologies.com/Ontology1286286598.owl

[9] http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#dfn-typed-literal

- An anonymous individual related through an object property to some resource or through a datatype property to a literal value. In this case, an anonymous resource is related through the property to the resource or literal, and the anonymous individual is not related to anything.
- A datatype property that has as range an enumerated datatype (i.e., an enumeration of literals). In this case, an *owl:Datatype* class is created as well as an anonymous individual of type *owl:Datatype*. However, the *owl:Datatype* class does not exist in the current specifications; only *rdfs:Datatype* does.

There is another case in which the test execution fails; when the ontology imports another ontology, the OWL API does not produce any ontology. This happens because, as the OWL API cannot find the ontology in the *owl:imports* property, it does not import anything. However, the tool should not rely on having full access to an ontology for just importing such ontology.

After analysing the results of the **NeOn Toolkit** we obtain the same conclusions as those previously presented for the OWL API with one exception; when the ontology contains an anonymous individual related to another anonymous individual through an object property, the execution of the NeOn Toolkit fails.

When **Protégé OWL** processes OWL DL ontologies, it usually produces the same ontology. The cases in which the ontologies are different occur when the ontology contains

- A literal value. The literal value is created with a datatype of *xsd:string* and, therefore, it is a different literal. As mentioned above, one requirement for literals to be equal is that either both or neither have datatype URIs.
- Class descriptions that are the subject or the object of an *rdfs:subClassOf* property. In these cases, the class description is defined as equivalent to a new class named "Axiom0"; this new class is the subject or the object of the *rdfs:subClassOf* property.

6 Interoperability Results

This section presents the interoperability results of the six tools evaluated. Table 2 presents the tool interoperability results for RDF(S), OWL Lite and OWL DL, respectively[10].

The tables show the percentage of tests in which the original and the resultant ontologies involved 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.

The conclusions of the behaviour of the tools that can be obtained from the interoperability results are the same as those already presented when analysing their conformance. The only new fact obtained while analysing the interoperability results stems from the interchanges of OWL DL ontologies from the OWL API (or from those tools

[10] As in the conformance tables, in these tables we have not counted additions of *owl:NamedIndividual* and *owl:Ontology*.

Table 2. Interoperability results.

(a) RDF(S).

	JE	SE	PO	NT	OA	P4
JE	100	100	83	0	0	0
SE	100	100	83	0	0	0
PO	83	83	83	0	0	0
NT	0	0	0	0	0	0
OA	0	0	0	0	0	0
P4	0	0	0	0	0	0

(b) OWL Lite.

	JE	SE	OA	P4	NT	PO
JE	100	100	98	98	95	89
SE	100	100	98	98	95	89
OA	98	98	98	98	95	89
P4	98	98	98	98	95	89
NT	95	95	95	95	95	87
PO	89	89	89	89	87	89

(c) OWL DL.

	JE	SE	OA	P4	NT	PO
JE	100	100	98	98	98	76
SE	100	100	98	98	98	76
OA	98	98	98	98	98	75
P4	98	98	98	98	98	75
NT	98	98	98	98	98	75
PO	76	76	75	75	75	76

that use the OWL API, i.e., Protégé 4 and the NeOn Toolkit) to Protégé OWL. In these interchanges, when the ontology contains an anonymous individual related through a datatype property to a literal value, Protégé OWL has execution problems.

7 Conclusions

This paper has presented an overview of the problem of interoperability between semantic technologies, first from a theoretical perspective and, second, by evaluating some tools and describing some of the problems the tools encounter when processing and exchanging ontologies.

In the results we can observe that all the tools that manage ontologies at the RDF level (Jena and Sesame) have no problems in processing ontologies regardless of the ontology language. Since the rest of the tools evaluated are based in OWL or in OWL 2, their conformance and interoperability is clearly better when dealing with OWL ontologies.

From these results we can also note that conformance and interoperability are highly influenced by development decisions. For example, the decision of the OWL API developers (propagated to all the tools that use it for ontology management) of converting all the ontologies into OWL 2 makes the RDF(S) conformance and interoperability of these tools quite low.

The results also show the dependency between the results of a tool and those of the ontology management framework that the tool uses; using a framework does not isolate a tool from having conformance or interoperability problems.

However, using ontology management frameworks may help increase the conformance and interoperability of the tools, since developers do not have to deal with the problems of low-level ontology management. Nevertheless, as observed in the results,

this also requires to be aware of the defects contained in these frameworks and to regularly update the tools and thus use their latest versions.

Acknowledgements

This work has been supported by the SEALS European project (FP7-238975). Thanks to Rosario Plaza for reviewing the grammar of this paper.

References

1. IEEE-STD-610: ANSI/IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology. IEEE (1991)
2. Duval, E.: Learning technology standardization: Making sense of it all. International Journal on Computer Science and Information Systems 1 (2004) 33–43
3. Sheth, A.: Changing focus on interoperability in information systems: From system, syntax, structure to semantics. In: Interoperating Geographic Information Systems. Kluwer (1998) 5–30
4. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer-Verlag (2007)
5. Brachmann, R., Levesque, H.: A Fundamental Tradeoff in Knowledge Representation and Reasoning. In: Readings in Knowledge Representation. Morgan Kaufmann, San Mateo (1985) 31–40
6. García-Castro, R., Grimm, S., Schneider, M., Kerrigan, M., Stoilos, G.: D10.1. Evaluation design and collection of test data for ontology engineering tools. Technical report, SEALS Project (2009)
7. García-Castro, R.: Benchmarking Semantic Web technology. Volume 3 of Studies on the Semantic Web. AKA Verlag – IOS Press (2010)
8. García-Castro, R., Toma, I., Marte, A., Schneider, M., Bock, J., Grimm, S.: D10.2. Services for the automatic evaluation of ontology engineering tools v1. Technical report, SEALS Project (2010)