SOMET: Algorithm and Tool for SPARQL Based Ontology Module Extraction

Paul Doran, Ignazio Palmisano, Valentina Tamma
pdoran, i.palmisano, valli@liverpool.ac.uk

Department of Computer Science, University of Liverpool

Abstract. The different ontology module extraction techniques need drawing together under a common framework, to allow for the selection, adaptation and combination of existing techniques. The current work upon a common framework places a significant overhead upon the user. The SOMET algorithm and tool uses the W3C standards, RDF and SPARQL. This allows the selection, adaptation and combination to be a simple manipulation of a SPARQL query. SOMET replicates the behavior of some of the existing techniques and there is work in progress in approximating approaches not fully captured. A preliminary evaluation of the efficiency of these techniques within SOMET is also conducted. A discussion is presented highlighting the advantages and disadvantages of SOMET.

1 Introduction

There are a plethora of techniques for carrying out ontology module extraction [1–5]. All of these techniques are designed for different applications and contain different assumptions about the problem as whole. Thus, there is a need to draw this work together within a common framework; the key advantage of a common framework is the ability to select, adapt and combine the different approaches. Also, development of new techniques is made easier, since common technical issues, such as whether reasoning is used or not, are already tackled in the framework and no effort is required by the developer.

The notion of a common framework is not new and has already been raised by d'Aquin *et al*[6] and Borgida and Giunchiglia[7]. [6] propose that the common framework be based upon graph transformations, whereby the ontologies are represented as a graph. In contrast [7] propose the use of a Tell/Ask interface.

The drawback of these approaches is that there could be a substantial overhead in adapting and combining the existing ontology module extraction approaches within their frameworks. [6]'s use of graph transformations requires Semantic Web practitioners and Ontology Engineers to become familiar with a new technique, which currently has no standardised syntax. Whilst the Tell/Ask interface used by [7] may be more familiar, since there is the DIG Interface [1] for

[1] http://dig.sourceforge.net/

Description Logic, there can be a significant overhead in adding functionality to the interface.

As such this paper proposes to use SPARQL[8] and RDF[9] as the common framework, both of which are W3C[2] standards. All OWL ontologies can be represented as an RDF graph and SPARQL is a query language for RDF. Thus, this paper shows how it is possible to cast the traversal based ontology module extraction approaches as a series of SPARQL queries upon an RDF graph. Note that adapting and combining the different ontology module extraction techniques only requires the manipulation of SPARQL queries. A more detailed explanation of how this can be accomplished is in Section 4.

This has been implemented in SOMET, which is available as a stand-alone tool. The tool replicates the behavior of the current traversal based extraction methods. Mechanisms for allowing the user to add their own queries to the extraction process are also available. Thus, SOMET achieves the aim of providing a common framework in which to represent these approaches. This is done within an environment with which most Semantic Web practitioners and Ontology Engineers are familiar with.

In Section 2 the related work is presented. Section 3 introduces RDF and SPARQL. Then Sections 4 and 5 present the SOMET framework and algorithm respectively. SOMET is evaluated in Section 6. A discussion is presented in Section 7. Lastly, Section 8 concludes.

2 Related Work

The literature on ontology modularization can be divided into two categories: ontology partitioning and ontology module extraction. Ontology partitioning [10, 11] divides an input ontology into a number of partitions, not necessarily disjoint. In contrast, ontology module extraction[1–5] takes an input ontology and extracts an ontology module about a supplied signature. Each approach has a bias in how to extract an ontology module based upon their assumptions and standpoint.

Doran, Tamma and Iannone[3] focus on extracting an ontology module that describes a single, user supplied, concept for the purpose of ontology reuse. Their approach is agnostic with respect to the language the ontology is expressed in, as long as the ontology can be transformed into their *Abstract Graph Model for Ontology Module Extraction*. The traversal carried out for extraction is done conditionally, with the conditions changing to suit the language that the ontology is expressed in. For example, if the start concept is involved in a *disjoint* relation then this will not be traversed.

d'Aquin, Sabou and Motta[2] present an extraction process which forms part of a knowledge selection process. The constraints posed by this process prioritise a reduction in the size of the module produced. To achieve this not all the superclasses of the included classes are included in the module. Shortcuts are taken in

[2] http://www.w3.org/

the class hierarchy by only including the most specific common super-classes (by applying the LCS, Least Common Subsumer[12], restricting the possible values for LCS to the classes already existing in the ontology).

Seidenberg and Rector[5] developed a technique specifically for the Galen[3] ontology, but it is possible to take the generic core and apply it to any ontology. It takes one or more classes of the ontology as an input, and anything that participates (even indirectly) to the description (and so the definition) of an included class has to be included. To reduce module size without losing relevant information, the authors filter Galen properties on the base of the property hierarchy.

Noy and Musen's[4] approach is based upon the notion of *traversal view extraction*. It is made available via the PROMPT[13] plugin for Protégé. Starting from one class of the considered ontology, this approach traverses the relations of this class recursively to include related entities. It can be distinguished from the other approaches in that it is intended as an interactive tool. This allows the user to incrementally build ontology modules by extending the currently extracted one. The user is allowed to select the relations to be traversed and associates to each of them a depth at which they should stop being traversed.

To some extent it is possible to cast [2, 3, 5] as instantiations of [4]. However, [2, 3] cannot be completely captured by the functionalities of [4]. [4] do not allow for the collapsing of the hierarchy, as used by [2]; or for the non-traversal of certain relations at specific points, as required by [3]. However, despite this all of these approaches can be seen as *traversal approaches*.

The orthogonal approach to the traversal approaches is the *logical approach*.

Cuenca-Grau *et al* [1] define a module as a minimal, conservative extension [14] of the original ontology with respect to the considered sub-vocabulary. For Cuenca-Grau *et al* [1] a module is that if the new axioms that are added to the original ontology are a conservative extension[14] of the original signature, then the original ontology (not the extension) is said to be a module of all the axioms together. Being a conservative extension means that the meaning of the terms in the input sub-vocabulary is completely captured by the module, as it is in the source ontology. In [1], Cuenca-Grau *et al* also show that computing a minimal module considering the definition they provide is undecidable, and describe two different approximations based on the notion of locality. The first method makes use of a reasoner to check the semantic locality of the axioms to be included. This procedure is decidable. The second one syntactically tests the locality of axioms and can be realised in polynomial time. This approach is not currently included in SOMET, but doing so is a focus of the future developments.

There has been some preliminary work on providing a common framework for ontology module extraction. d'Aquin *et al*[6] propose to use graph transformations as a common framework. An abstract model for representing ontologies as attributed graphs is described, along with the reformulation of existing module extraction techniques as graph transformation rules. The aim being to produce a parametric modularization tool. The approach presented by Borgida and

[3] http://www.opengalen.org/

Giunchiglia[7] can also be viewed as a step towards a common framework. They provide a Tell/Ask interface for the functional importing of knowledge bases.

3 RDF Graph and SPARQL

The Resource Description Framework (RDF) is a W3C standard which provides a common syntax and semantics for metamodelling of data. The semantics of RDF can be represented as a graph. An RDF graph (a labelled, directed graph) is a set of RDF triples. In RDF triples represent <*subject*, *predicate*, *object*> relations. For example, consider the following set of triples.

$$< C,\ \mathit{referees},\ \_ : x >$$
$$< A,\ \mathit{playsIn},\ \_ : x >$$
$$< B,\ \mathit{playsIn},\ \_ : x >$$

These triples are also a valid RDF graph and can be represented by the graph shown in Figure 1. ($\_ : x$ in the above is a blank node[4])

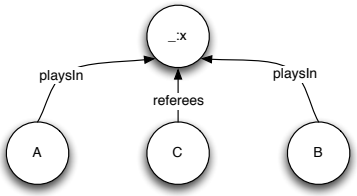

Fig. 1. RDF graph representation

As every valid OWL ontology is also a valid RDF graph[5] it is possible to use this graph as a common basis to build upon. This also brings the added benefit of being able to use SPARQL.

SPARQL is a W3C standard for querying RDF. SPARQL allows graph patterns to be queried along with their conjunctions and disjunctions. SPARQL query results can either be result sets or RDF graphs. For example, consider the following SPARQL query[6] based upon the triples presented before.

SELECT $?x$ WHERE { $?z$ *referees* $?x$ }

This query will return all $?x$ where the given pattern is matched. Following from the example the result for this query will be C. It should be noted that it is possible that $?z$ could be satisfied by multiple resources.

[4] http://www.w3.org/TR/rdf-concepts/#dfn-blank-node
[5] http://www.w3.org/TR/owl-semantics/mapping
[6] The PREFIX part of the query has not been included to improve readability.

However, for SOMET we need to be able to return an RDF graph. The query shown below will return an RDF graph of the results.

CONSTRUCT { ?*z referees* ?*x* } WHERE { ?*z referees* ?*x* }

The ability to return an RDF graph as a result of a query is important for SOMET as it allows for the iterative construction of an ontology module.

4 SOMET Framework

The SOMET framework is shown in Figure 2. The different approaches can be assimilated to a set of SPARQL queries. This is extensible as it allows the user to add/remove SPARQL queries from the set of queries that will be passed to the Traversal Extraction Engine. This flexibility is important because it allows the user to tailor the extraction process to their specific needs. This benefit is further enhanced due to the fact that the queries are represented in SPARQL, which is a standard that Semantic Web practitioners and Ontology Engineers are familiar with.

The different ontology module extraction techniques are represented as a set of SPARQL queries. These sets are not disjoint and so their intersections highlight the areas in common between the techniques.

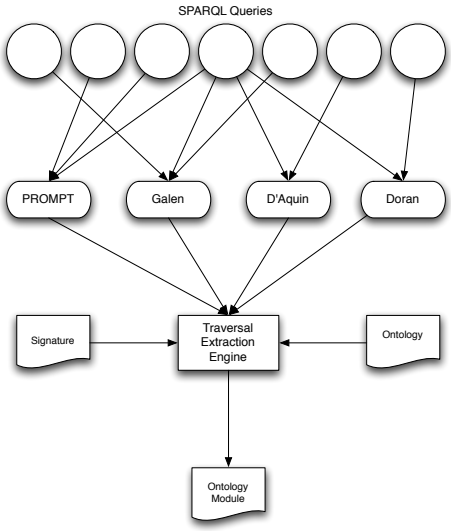

Fig. 2. SOMET Framework

The Traversal Extraction Engine (TEE) is implemented in Java and makes use of Jena[7], a programmatic environment for RDF, OWL and SPARQL; and Pellet [15], a OWL-DL reasoner. The user supplies the extraction engine with an ontology to extract an ontology module from and a signature that the ontology module should be about. From this the Traversal Extraction Engine is able to produce an ontology module by applying the algorithm presented in the following section.

Since the description of a strategy is broken down in a set of SPARQL queries, building a hybrid technique is simple. Each SPARQL query is represented by a ExtractionQuery object, which contains the query template, the number of arguments the query accepts and whether or not the arguments are allowed to be blank nodes. The set of queries is then the input to the TEE, along with the ontology. In order to add the characteristics of different approaches to an existing one, new ExtractionQuery objects must be created and added to the set used to invoke the TEE.

The TEE also has an optional filtering step, where the resulting module is reduced in size by running another set of SPARQL queries (in this phase limited to SELECT queries). Any resource matched by the queries is then removed from the resulting module.

5 SOMET Algorithm

The algorithm used by SOMET is a paramaterised graph traversal algorithm, see Algorithm 1. The parameters to the algorithm are the ontology you wish to extract a module from, a signature to describe the module and a set of SPARQL queries. The algorithm iteratively applies the queries to the elements of the signature to construct a module; any element returned by a query that is not in the signature also has the queries applied to it. By doing this the ontology module is iteratively constructed.

5.1 Breakdown of SPARQL queries used.

This section will present the queries used by the techniques currently implemented in the SOMET framework. A brief description will be provided for each query; %1$s present in these descriptions represents the current resource of focus[8]. This section does not list all the queries used, due to space limitations, but a full list can be found at http://www.csc.liv.ac.uk/~pdoran/SOMET/queries.html.

Doran, Tamma and Iannone [3] present an approach focused on extracting an ontology module about a single concept, with the aim that this ontology module can be reused. The queries used are:

[7] http://jena.sourceforge.net

[8] this syntax is the Java way of representing an argument which must be replaced with a string. http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html#format

Algorithm 1 SOMET Graph Traversal Algorithm for Ontology Module Extraction

INPUT

- O **- An RDF Graph (**O=(V,E)**)**
- S **- a signature ; where** $S \subseteq V \cap E$
- Q **- a set of SPARQL queries.**

OUTPUT

- M **- An RDF Graph**

procedure *graphTraversalExtraction*(O, S, Q)
Visited - a container of V **that have been visited**
ToVisit - a container of V **to be visited.**
insert S **into ToVisit**
while ToVisit is not empty do
  x **= first element of ToVisit**
  remove x **from ToVisit**
  if $x \notin$ **Visited then**
    for all $q \in Q$ **do**
      r **= apply** q **to** O
      subjects in r **are added to ToVisit**
      $M = M +$ **r**
    end for
  end if
end while

DESCRIBE %1$s

Describes the current resource. The DESCRIBE keyword definition in the SPARQL Recommendation is labeled as "informative", which means the specific SPARQL implementation answering the query has some degree of freedom about what to consider part of the description of a resource. We use ARQ[9], which implements the DESCRIBE as a list of the statements in which the resource appears as subject, plus the closure computed from any blank nodes involved.

CONSTRUCT {?y rdfs:domain %1$s.} WHERE {?y rdfs:domain %1$s.}
Returns all the ?y where the resource of interest is the domain.

DESCRIBE ?y WHERE {?y rdfs:subClassOf %1$s.}
Returns the subclasses of the current resource.

CONSTRUCT {?y owl:equivalentClass %1$s.} WHERE {?y owl:equivalentClass %1$s.}
Returns all the ?y that the current resource that are owl:equivalentClass

[9] http://jena.sourceforge.net/ARQ/

d'Aquin *et al* [2] present an approach requiring the extraction of a module for the dynamic scenario of knowledge selection. Our implementation includes the implementation of a LCS operator as a ARQ extension[10]. In a similar way, MSC and most specific property ancestor are implemented, as *property functions*.

It is important to note that this approach requires that the input Jena Model containing the ontology be backed by a reasoner, in order to get correct results; in fact, subsumption is necessary to correctly compute the LCS.

An example of how to use the extensions is in the following query:

```
DESCRIBE ?msc
WHERE {GRAPH %2$s{{{?a a owl:Class}
FILTER(isURI(?a) && !sameterm(?a,%1$s))}
UNION{{?a a rdfs:Class}
FILTER(isURI(?a) && !sameterm(?a,%1$s))}}
{?msc ¡java:sparking.propertyfunctions.lcs¿ (?a %1$s).}}
```

It returns a graph containing the definition of the class that is the LCS that subsumes the current class and another class, which is already included in the module (this condition is enforced by the GRAPH instruction, which uses the support to Named Graphs in SPARQL to restrict the solutions to a subset of the whole ontology).

Seidenberg and Rector [5] present an approach focused on extracting a module from Galen, but the core of the technique can be applied to any ontology. The queries for subclasses and equivalences are similar to those used in the Doran, Tamma and Iannone queries.

Our replication of this approach includes a filtering step, in which a set of properties definition is discarded, as in the original approach:

```
SELECT ?y WHERE {{?y rdfs:subPropertyOf
  ¡http://www.co-ode.org/ontologies/galen#FunctionalAttribute¿}"
```

This query selects any subproperty of http://www.co-ode.org/ontologies/galen#FunctionalAttribute; the TEE will then remove those definitions from the module.

Noy and Musen [13] present an approach for extracting a module based on the notion of view and this approach is highly user configurable. The query used is:

```
CONSTRUCT { ?x %2$s %1$s }
WHERE { ?x %2$s %1$s }
```

As PROMPT is user configurable the query is also user configurable. %$1s remains the current resource, whilst we introduce a second parameter (%$2s) to represent the predicate. The traversal depth becomes an extra parameter to the traversal algorithm, which needs to be passed to PROMPT for processing.

In SOMET, we don't attempt to reimplement the whole of PROMPT so far; we plan to do so in future developments.

[10] ARQ extensions are explained at http://jena.sourceforge.net/ARQ/extension.html

6 Preliminary Evaluation

There are two aspects to consider in evaluating SOMET, these are:

1. What can be said about these techniques in terms of efficiency?
2. Does SOMET replicate the existing ontology module extraction techniques?

The first aspect of the evaluation is difficult. There is a need for the ontology modularisation community as whole to address the issue of evaluation. Existing evaluation criteria used to measure the 'quality' of the modules produced are insufficient; however, this point is beyond the scope of this paper. Considering this we evaluated how well the different techniques performed on the SOMET framework. These results are shown in Table 1. In this table we report running time average (in milliseconds) and triple size average for the three approaches. For the three ontologies presented here, we ran a different number of tests: 50 on the Galen fragment, 48 for Mind.owl and 285 for Portal.owl; in the last two cases, these are the number of classes in the ontologies, while for Galen we limited the test to only 50 of the 4500 classes in there (choosing a random set of concepts), due to the average time requested to run on this larger ontology.

	Seidenberg time avg	**Doran time avg**	**d'Aquin time avg**	**Seidenberg triple avg**	**Doran triple avg**	**d'Aquin triple avg**
Galen fragment	14502.4	46938.42	880.28	36719.88	43834.44	676.88
Mind.owl	152.4	247.94	165.67	285.88	621.33	170.35
Portal.owl	2560.94	5563.13	2055.17	6128.49	4668.31	4881.48

Table 1. Table showing performance of the different techniques on the SOMET framework.

In Figure 3, the performance of SOMET on the Galen fragment is depicted graphically; in Figure 4, we report the triple size of the extracted modules (here, *Terms* refers to the starting points for each extracted modules). The data in Table 1 shows that the d'Aquin *et al* approach on average is quicker and produces smaller modules; this is highlighted further by Figures 3 and 4. However, this does not say anything about the quality of the modules produced by the different approaches. Looking across the data as a whole it is possible to see that there are certain 'zones', this would suggest that these 'zones' conform to a coherent module. This aspect requires further investigation.

The second aspect of the evaluation requires an in-depth set of experiments to prove that the SOMET framework fully captures all the functionalities of the different ontology module extraction techniques. This has been completed for Seidenberg and Doran approach, over Galen (Seidenberg implementation is targeted at this ontology, therefore validating using different ontologies is hard). Further experiments on d'Aquin approach are ongoing at the time of writing;

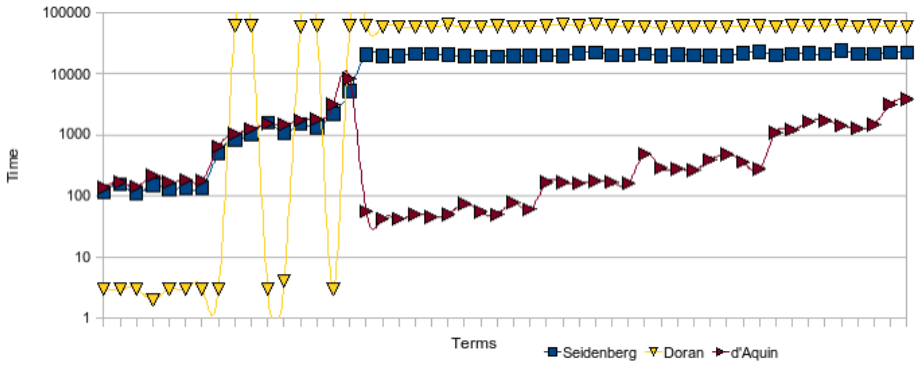

Fig. 3. Time performances on Galen fragment

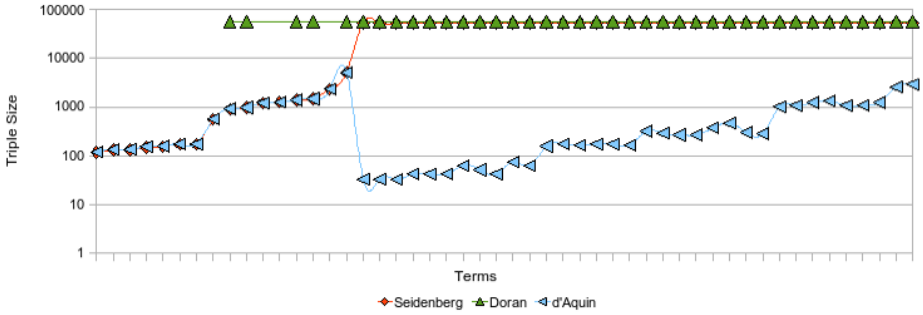

Fig. 4. Triple size of the extracted modules

we do not report them here since including a partial set of possibly inconclusive results would not be appropriate.

7 Discussion

A common framework provides a fair basis for comparing the different ontology module extraction techniques. The ability to compare the techniques in this way is important because it will allow an Ontology Engineer to discriminate objectively between the different techniques. Indeed the user can try the different techniques in a convenient manner.

Whilst a common framework is an important step forward it further highlights the need for robust mechanisms for evaluation. Current attempts at an objective evaluation based on size, or precision and recall [3] are not optimal.

The precision and recall metrics applied by [3] only consider the taxonomic structure and, as such, Doran *et al* state that these need extending to capture the more expressive elements of an ontology. This is a focus of current research.

The aim of modularization in general is to reduce the size of an ontology, but this is not an end in itself because it introduces the obvious paradox that the optimum module size is 0. Whilst an ontology module of size 0 would be highly reusable and the effort required negligible, it is evident that it is also useless, therefore size must be traded off with some other metric. Finding such objective metric is not a straightforward task; for a comprehensive overview of the literature on ontology evaluation see [16].

8 Conclusion and Future Work

SOMET provides a common framework, based on RDF and SPARQL, for the selection, adaptation and combination of different ontology module extraction techniques. It was shown that it is possible to replicate the current techniques as SPARQL queries. Even difficult aspects of certain techniques, such as the most specific concept, could be implemented. The preliminary evaluation conducted allowed us to compare the efficiency of the different techniques, both in terms of time and the size of the modules produced. However, this preliminary evaluation highlighted the need for more robust objective criteria to compare the quality of the modules produced.

Future work includes assimilating the logical approaches, such as [1] into the SOMET framework. This is an important step that needs to be taken. In addition, there is a need to investigate more robust mechanisms for evaluating the quality of a module. This is because the current objective criteria used are not powerful enough.

Acknowledgements This work was supported by the Engineering and Physical Sciences Research Council (EPSRC).

References

1. Cuenca-Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: Extracting modules from ontologies. In: Proc. of the 16th International World Wide Web Conference (WWW 2007), pages 717-727, Banff, AB, Canada, May 2007. ACM Press. (2007)
2. d'Aquin, M., Sabou, M., Motta, E.: Modularization: a key for the dynamic selection of relevant knowledge components. In: First International Workshop on Modular Ontologies, ISWC 2006, First International Workshop on Modular Ontologies, ISWC 2006, Athens, Georgia, USA. (2006)
3. Doran, P., Tamma, V.A.M., Iannone, L.: Ontology module extraction for ontology reuse: an ontology engineering perspective. In Silva, M.J., Laender, A.H.F., Baeza-Yates, R.A., McGuinness, D.L., Olstad, B., Olsen, Ø.H., Falcão, A.O., eds.: CIKM, ACM (2007) 61–70
4. Noy, N.F., Musen, M.A.: Specifying ontology views by traversal. In: International Semantic Web Conference. (2004) 713–725
5. Seidenberg, J., Rector, A.: Web ontology segmentation: analysis, classification and use. In: WWW '06: Proceedings of the 15th international conference on World Wide Web, New York, NY, USA, ACM Press (2006) 13–22
6. d'Aquin, M., Doran, P., Motta, E., Tamma, V.: Towards a parametric ontology modularization framework based on graph transformation. [17]
7. Borgida, A., Giunchiglia, F.: Importing from functional knowledge bases - a preview. [17]
8. SPARQL: Query language for RDF. W3C Recommendation - 15th January 2008
9. RDF: Resource description framework RDF. W3C Recommendation - 10th February 2004
10. Cuenca-Grau, B., Parsia, B., Sirin, E., Kalyanpur, A.: Automatic partitioning of owl ontologies using e-connections. In: Proceedings of the 2005 International Workshop on Description Logics (DL-2005). (2005)
11. Stuckenschmidt, H., Klein, M.: Structure-based partitioning of large concept hierarchies. In: Proceedings of the 3rd International Semantic Web Conference, Hiroshima, Japan (2004)
12. Cohen, W.W., Borgida, A., Hirsh, H.: Computing Least Common Subsumers in Description Logics. In: AAAI. (1992) 754–760
13. Noy, N.F., Musen, M.A.: Prompt: Algorithm and tool for automated ontology merging and alignment. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, AAAI Press / The MIT Press (2000) 450–455
14. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In Veloso, M.M., ed.: IJCAI. (2007) 453–458
15. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics **5**(2) (2007)
16. Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J.: Ontology evaluation and validation: an integrated formal model for the quality diagnostic task. Technical report, Laboratory for Applied Ontology (2005)
17. Cuenca-Grau, B., Honavar, V., Schlicht, A., Wolter, F., eds.: Proceedings of the 2nd International Workshop on Modular Ontologies, WOMO'07, Whistler, Canada, October 28, 2007. (2007)