OWL Instance Data Evaluation

Li Ding, Jiao Tao, and Deborah L. McGuinness

Tetherless World Constellation, Computer Science Department
Rensselaer Polytechnic Institute
110 8th st., Troy, NY 12180
{dingl, taoj2, dlm}@cs.rpi.edu

Abstract. As more applications are depending on semantic web data from diverse sources, semantic web data evaluation is becoming more critical. While language validators and general reasoners can help, these typically focus on syntax and logical consistency. Many applications need additional support for finding possible issues (as well as provable mistakes). We are investigating methods and environments that provide computational support for identifying possible problems with instance data. We report on a line of work focusing on evaluation of provable and possible problems with OWL instance data and provide some discussion motivated by our first application setting validating large amounts of diverse explanation data.

Keywords: Semantic Web, OWL instance data evaluation

1 Introduction

Before consuming semantic web data, many applications and users deploy automated tools to find problems early. Typically users will deploy a syntax checker, such as the W3C RDF Validator. They may also deploy tools such as Pellet to check semantic consistency. These tools typically provide techniques aimed at finding provable (rather than possible) issues. We present two examples using the wine ontology[1] that motivate instance data evaluation going beyond what conventional tools provide.

1. When creating an instance W1 of wine:Zinfandel, a user may forget to provide information about the maker of the wine. Before using W1, a warning may be beneficial concerning the missing value since the wine ontology asserts that each wine has exactly one maker. OWL reasoners will not produce an error message about this point because they use the open world assumption, and believe that maker information will be provided later.
2. When creating an instance W2 of wine:Zinfandel, a user may mistakenly assert that the maker of W2 is wine:Red, whose type is wine:WineColor. A warning message may be beneficial because the wine ontology states that the range of wine:hasMaker is wine:Winery. OWL reasoners will infer that wine:Red is an instance of wine:Winery, and they will not generate an error message unless they

[1] The wine ontology (http://www.w3.org/TR/owl-guide/wine) is provided in the OWL guide.

know wine:Red can not be an instance of wine:Winery e.g., because wine:WineColor is owl:disjointWith wine:Winery.

Previous work related to semantic web data evaluation mainly focuses on general ontology evaluation [1-4] and OWL DL ontology debugging [5-10]. None of these efforts directly focused on OWL instance data and the resulting tools do not satisfy our motivating examples. Therefore, more comprehensive OWL instance data evaluation method and environment work is needed.

Our work focuses on web settings where applications ingest large amounts of OWL instance data. We are developing an extensible OWL instance data evaluation environment that inspects instance data for provable errors as well as possible issues, such as compatibility with the term descriptions in the corresponding ontologies or conceptual modeling style guidelines. Our initial contributions include a list of issues and system designs for OWL instance data evaluation, and preliminary results utilizing SPARQL in our implementation.

2 Issues in OWL Instance Data Evaluation

We will describe the data to be evaluated and the associated evaluation services.

Scope of data. Semantic Web data represented in OWL can be viewed as an RDF graph, i.e. a set of RDF triples; therefore, our environment basically checks a set of RDF triples describing individuals for issues that need further study. We use an RDF document (i.e. a document serializing some RDF triples) to provide an operational boundary for RDF triples to be evaluated by our environment. We can refine the selection of RDF triples by grouping RDF triples[2] into (i) ontology (triples describing known classes/properties), (ii) instance (triples describing known individuals) and (iii) triples which do not belong to ontology or instance data.

Evaluation service process. Our extensible workflow for OWL instance data evaluation consists of the following steps:

1. Load an RDF document.
2. Parse OWL instance data from the RDF document using a RDF syntax validation service such as the Jena RDF parser.
3. Load the corresponding ontologies[3] used by the OWL instance data.
4. Inspect the OWL instance data for issues such as consistency and compatibility issues.

While steps 1-3 can be implemented using existing tools, step 4 involves a wide spectrum of evaluation services and should be customizable to meet different application requirements. Currently, we focus on issues related to compatibility e.g. whether the type of an individual is compatible with the expected type suggested by the corresponding ontologies, and we assume that the corresponding ontologies are available and free of semantic inconsistency.

---

[2] Triples can be grouped by their subject RDF resources, which can be classified as class, property, individual, and untyped according to OWL semantics. Triples may appear in multiple groupings if working with OWL-Full data.

[3] We focus on online OWL ontologies which are shared and accessed on the Web.

Issues related to an individual's type description:

- Unexpected individual type: the known types of an individual do not meet the restrictions specified by domain, range, and/or property value restrictions in the corresponding ontology.
- Redundant class/property instantiation, e.g. an individual is typed as a class C and typed as C's super-classes in explicit (non-inferred) statements.
- Non-leaf class recognition for an individual. This is triggered when an individual is recognized to be an instance of a class that has subclasses, but the individual is not recognized to be an instance of any of the subclasses.

Issues related to property restrictions:

- Missing property values: more property values are needed to meet the minimum property cardinality restrictions.
- Missing individual (in)equality information for confirming whether the maximum property cardinality restrictions have been satisfied.

To detect the above issues, we are investigating a SPARQL based approach. We assume the existence of an OWL DL reasoner with a SPARQL query interface such that we can run SPARQL queries on the triples inferred by the reasoner that has access to instance data and the associated OWL ontologies. With this assumption, first, we write several SPARQL queries to inspect each of the issues. Then each query is run against the inferred RDF dataset generated by the assumed reasoner. We encode the first example in the introduction in http://tw.rpi.edu/2008/03/wine-instance.owl which has three instances of wine:Zinfandel (with W3 missing a value for wine:hasColor), and use http://tw.rpi.edu/2008/03/wine.owl, which is a fragment of the wine ontology. We show a SPARQL query (see below) for detecting the issue of "missing property values". The query does not return the wine W3 because the definition of wine:Zinfandel in the wine ontology already asserted the value of hasColor using an owl:hasValue restriction; and it only returns the wine W1 which does not meet the expectation of the owl:cardinality restriction on the property hasMaker (the restriction is inherited from wine:Wine).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?w   ?p   ?g_i
FROM NAMED <http://tw.rpi.edu/2008/03/wine-instance.owl>
FROM NAMED <http://tw.rpi.edu/2008/03/wine.owl>
WHERE { GRAPH ?g_o {   ?c rdfs:subClassOf ?r.
                       ?r rdf:type owl:Restriction .
                       ?r owl:onProperty     ?p .
                       ?r owl:cardinality ?card .
                       FILTER( ?card = 1 )
                       OPTIONAL {   ?c rdfs:subClassOf ?r1.
                                    ?r1 rdf:type owl:Restriction .
                                    ?r1 owl:onProperty     ?p .
                                    ?r1 owl:hasValue ?val . }
                       FILTER( !BOUND(?r1) )        }
        GRAPH ?g_i {   ?w rdf:type ?c .
                       OPTIONAL { ?w ?p ?o . }
                       FILTER( !BOUND(?o) ) }   }
```

3 Conclusion

Inspired by our past experiences on ontology evolution environments [3] and our current need to evaluate large amounts of disparate instance data, we are developing a comprehensive web based OWL instance data evaluation environment. We will include evaluation services concerning provable issues such as OWL DL semantic consistency checking, as well as possible issues such as checking if an instance meets the expected cardinality restrictions from the corresponding ontologies. Our tools for checking possible issues will support users who need additional support in bridging the open world reasoning paradigm of description logics with the closed world reasoning paradigm of database integration. We are also designing APIs that let users add their own tools to check for application- or environment-specific requirements such as special naming conventions and style conventions. We are running our tests over OWL instance data such as Proof Markup Language explanation metadata in our Inference Web project [11]. We also plan to test our implementation using more real world datasets collected by tools such as Swoogle [12].

References

1. Y. Sure, G. Gomez-Perez, W. Daelemans, M. Reinberger, N. Guarino, and N. F. Noy. Why Evaluate Ontology Technologies? Because It Works! IEEE Intelligent Systems, 19(4): 74-81. (2004)
2. A. Gomez-Perez. Evaluation of Ontologies. Intl. Journal of Intelligent Systems, 16(3): 391-409. (2001)
3. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An Environment for Merging and Testing Large Ontologies. In KR, pp. 483-493. (2000)
4. M. Gruninger, M. S. Fox. Methodology for the Design and Evaluation of Ontologies. In Workshop on Basic Ontological Issues in Knowledge Sharing. (1995)
5. B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL Ontologies. In WWW, pp. 633-640. (2005)
6. P. Plessers and O. D. Troyer. Resolving Inconsistencies in Evolving Ontologies. In ESWC, pp. 200-214. (2006)
7. H. Wang, M. Horridge, A. Rector, N. Drummond, and J. Seidenberg. Debugging OWL-DL Ontologies: A Heuristic Approach. In ISWC, pp. 745-757. (2005)
8. K. Baclawski, C. J. Matheus, M. M. Kokar, J. Letkowski and P. A. Kogut. Towards a Symptom Ontology for Semantic Web Applications. In ISWC, pp. 650-667. (2004)
9. Ó. Corcho, A. Gómez-Pérez, R. González-Cabero, and M. del Carmen Suárez-Figueroa. ODEVAL: A Tool for Evaluating RDF(S), DAML+OIL and OWL Concept Taxonomies. In AIAI, pp. 369-382. (2004)
10. S. Bechhofer and R. Volz. Patching Syntax in OWL Ontologies. In ISWC, pp. 668-682. (2004)
11. D. L. McGuinness and P. Pinheiro da Silva. Explaining Answers from the Semantic Web: The Inference Web Approach. Journal of Web Semantics, 1(4): 397-413. (2004)
12. L. Ding, T. Finin, A. Joshi, R. Pan, R. Scott Cost, Y. Peng, P. Reddivari, V. Doshi and J. Sachs. Swoogle: A Search and Metadata Engine for the Semantic Web. In CIKM, pp. 652-659. (2004)