Ontology Consistency and Instance Checking
For Real World Linked Data

Gavin E. Mendel-Gleason, Rob Brennan, and Kevin Feeney

Trinity College Dublin
{mendelgg, rob.brennan, kevin.feeney}@scss.tcd.ie

Abstract. Many large ontologies have been created which make use of OWL's expressiveness for specification. However, tools to ensure that instance data is in compliance with the schema are often not well integrated with triple-stores and cannot detect certain classes of schema-instance inconsistency due to the assumptions of the OWL axioms. This can lead to lower quality, inconsistent data. We have developed a simple ontology consistency and instance checking service, SimpleConsist[8]. We also define a number of ontology design best practice constraints on OWL or RDFS schemas. Our implementation allows the user to specify which constraints should be applied to schema and instance data.

1 Introduction

Many Linked Data stores have large amounts of quite variable[7] data (e.g. DBpedia[3]). Triples can exist in a triple-store which have no associated schema or conform to no constraints on the shape or type of data. Typically such data is considered low quality and is hard to consume.

Earlier work showed that OWL semantics make it ill suited as a language of constraints[9]. However maintaining ontology consistency and conformance is central to high quality data storage. Programmatic consumption of data is simplified if the data is well formed and well typed. Data management is simplified if inserts, deletes and updates that might violate well formedness constraints is signalled.

To solve these problems, we use a persistent triple-store in ClioPatria[1] and a plugin constraint checker called SimpleConsist[8], both implemented in SWI-Prolog[11]. SimpleConsist is used to maintain ontological consistency and constraints on instance data such that it conforms to an ontology described in an OWL fragment using a narrower reading of the OWL semantics. In particular, we make use of a *closed world assumption*, and a *unique names assumption*. It is implemented as a REST service within the Dacura[6] data curation system.

2 Ontology Consistency and Instance Checking

The philosophy for our ontology consistency and instance checking is to view the ontology as static assertions, which must be self-consistent, and to which a given instance state must conform. Given some triple-store state S we check to make sure that our set of constraints C are satisfied. When updating or inserting into the triple-store, somewhat arbitrary program logic can take place, after which the triple-store is in a state S'. If C does not hold for S' then we roll-back to the previous state S. We provide counter-example witnesses L to the failure of the constraint C. These witnesses are useful for debugging schema and instance updates as it gives information about what precisely went wrong in the constraint checking. The constraint rules are a combination of consistency constraints, instance type checking and best practices.

2.1 Constraints

SimpleConsist implements our constraints on ontology consistency and implements instance checking. Because we want witnessing information of the failure to satisfy the constraints, we write constraints which yield the witnesses of a failure. These witnesses are realised as resources not conforming to the constraints. Failure to provide a witness of the negation of the constraint is viewed as success. The failure witnessing predicates are briefly described in Table 1.

All witnesses of class cycles are given in the list `L`. Each element of the list names both the offending class, and the path through the classes. The other constraints return information about the reasons for failure.

`invalidInstanceRange(L)` requires some explanation as it is an implementation of a type checker for literals and class instances and so requires knowing what a literal can be. The constraint implements type checking to ensure that all literals are of the appropriate type according to the ranges specified in properties. These literals can be any RDF literal types of the XML Schema which are valid for OWL[5]. All ranges which specify a class have targets which are instances of an appropriate class (either the class itself or a subclass). Using artificially populated triple stores we timed the reasoner for various numbers of triples generated from the instance generator. These timings can be seen in Figure 1.

3 Prior Work

There are many reasoners for fragments of OWL, e.g 17 are mentioned in [10]. Many are sophisticated, however, the lack of the *unique name*

assumption can lead to problems for users developing schemata, making it virtually impossible to use OWL to impose constraints.

CWM (Closed World Machine)[4] is a reasoner which takes our same pragmatic approach to closed worlds and unique names. It is capable of expressing the types of constraints we are interested in, in a parsimonious fashion. However, it functions at the level of transformations of RDF files rather than being a fully functioning database system. Running the reasoner would require export of the triple-store which is not practical for large datasets which are changing in real-time.

There are several tools provided with the Apache Jena[2] system which facilitate consistency and instance checking. In particular the Eyeball system is modular and allows the user to introduce new constraints by adding Java code to perform inspection. However, it does not implement full type checking of instance data as our constraints do.

4 Future Work and Conclusion

Triple stores may be applied to complex ontologies which have incremental schema changes. However it is a challenge to provide tools which make publishing OWL-based high quality (consistent) large scale data easy. This requires constraints on schemata and admin tools to ensure that updates to datastores maintain integrity. Our SimpleConsist service has provided practical solutions to these problems. We found it useful to reduce the expressive complexity which can be found in OWL when constructing our interpretation of ontologies, limiting to *unique names* and *closed worlds* and preferring to allow higher level data curation processes deal with the greater ambiguity often inherent in large scale data.

In future work constraint checks on more OWL features will be explored. Our priority are OWL features that do not come into conflict with manageability of the schema and tractability of constraint checking. We would also like to have a method of checking instance updates which limits checking to entities which could cause constraint failures. Instance updates are generally more frequent than schema changes and so checker execution time will be more important.

5 Acknowledgement

This research is partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 644055 (ALIGNED, aligned-project.eu).

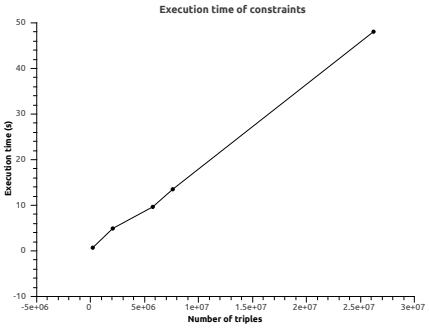

Fig. 1: Execution time of all constraints.

¬ `duplicateClasses(L)`	No two classes may have the same name.
¬ `orphanSubClasses(L)`	No subclass can be a child of an unspecified class.
¬ `classCycles(L)`	No cycles exist in the class hierarchy.
¬ `duplicateProperties(L)`	No two properties have the same name.
¬ `orphanSubProperties(L)`	No Subproperty is the child of an unspecified property.
¬ `propertyCycles(L)`	No cycles exist in the property hierarchy.
¬ `invalidRange(L)`	Ranges must refer to classes or types, and must be unique.
¬ `invalidDomain(L)`	Domains must refer to classes or types, and must be unique.
¬ `orphanInstances(L)`	Instances must be members of a class.
¬ `orphanProperties(L)`	Instances must not use properties which are not defined.
¬ `invalidInstanceRange(L)`	An element of the range of a property must be well typed.
¬ `invalidInstanceDomain(L)`	An element of the domain of a property must be well typed.

Table 1: List of constraints

References

1. Whitepaper: The ClioPatria semantic web server. http://cliopatria.swi-prolog.org/help/whitepaper.html.
2. Apache. Apache Jena, 2015.
3. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *In 6th International Semantic Web Conference, Busan, Korea*, pages 11–15, 2007.
4. T. Berners-Lee. CWM - closed world machine. http://www.w3.org/2000/10/swap/doc/cwm.html, 2000.
5. Jeremy J. Carroll and Jeff Z. Pan. XML Schema datatypes in RDF and OWL. W3c working group note, W3C, March 2006.
6. Kevin C. Feeney, Declan O'Sullivan, Wei Tai, and Rob Brennan. Improving curated Web-Data quality with structured harvesting and assessment. *Int. J. Semant. Web Inf. Syst.*, 10(2):35–62, April 2014.
7. Johannes Lorey, Ziawasch Abedjan, Felix Naumann, and Christoph Böhm. RDF ontology (Re-)Engineering through large-scale data mining. In *International Semantic Web Conference (ISWC)*, November 2011. Finalist of the Billion Triple Challenge.
8. Gavin Mendel-Gleason. SimpleConsist plugin for ClioPatria. https://github.com/GavinMendelGleason/dacura.
9. Boris Motik, Ian Horrocks, and Ulrike Sattler. Adding integrity constraints to OWL. In *OWLED*, volume 258, 2007.
10. Wei Tai, John Keeney, and Declan O'Sullivan. Resource-constrained reasoning using a reasoner composition approach. *Semantic Web*, 6(1):35–59, January 2015.
11. Jan Wielemaker, Tom Schrijvers, Markus Triska, and Torbjörn Lager. SWI-prolog. November 2010.