

# Redefining the RDFS Closure to be Decidable

Jesse Weaver

Tetherless World Constellation, Rensselaer Polytechnic Institute, Troy, NY, USA  
weavej3@cs.rpi.edu

## 1 Introduction

In this position paper, I review a problem with the way the current RDF Semantics [1] defines the RDFS closure. In particular, the concern is with the treatment of container membership properties which causes the RDFS closure to be infinite. This problem has been known for over half a decade [2]. I present some statistics on the usage of container membership properties that suggest that container membership properties are used widely enough to advocate continued support and thus modification to the definition of the RDFS closure. I look at some previously suggested solutions and then propose a solution which involves adding four rules to the RDFS entailment rules and removing some RDF and RDFS axiomatic triples.

## 2 Problem

The current recommendation of the RDF semantics states the *RDFS entailment lemma* as follows:

“*S* rdfs-entails *E* if and only if there is a graph which can be derived from ***S* plus the RDF and RDFS axiomatic triples** by the application of rule *lg*, rule *gl* and the RDF and RDFS entailment rules and which either simply entails *E* or is an XML clash.” [1]

As pointed out by ter Horst [2, 3], the problem is that the RDFS closure is defined to be an infinite number of triples. (This was also noted by Mika [4].) This is because the first step of computing the RDFS closure is to add the RDF and RDFS axiomatic triples, and these sets of triples contain the following set of infinite triples:

```
rdf:_1 a rdf:Property, rdfs:ContainerMembershipProperty ;
      rdfs:domain rdfs:Resource ; rdfs:range rdfs:Resource .
rdf:_2 a rdf:Property, rdfs:ContainerMembershipProperty ;
      rdfs:domain rdfs:Resource ; rdfs:range rdfs:Resource .
...
```

This requirement renders the problem of computing the complete RDFS closure undecidable.

## 3 Usage

Before considering solutions to the problem, one might consider whether support for container membership properties is even warranted as it may be a less desirable feature of RDF(S). To this end, I take a cursory look at some statistics on the usage of container membership properties in the 2009 Billion Triples Challenge dataset<sup>1</sup>.

<sup>1</sup> <http://vmlion25.deri.ie/>, last accessed April 5, 2010.

The 2009 Billion Triples Challenge dataset is a mixed quality dataset of approximately 1.14 billion quads crawled from the web. It contains 898,966,813 unique triples, of which 22,282,626 (nearly 2.5%) contain a container membership property (`rdf:_i`) as the predicate. To put that in perspective, if all the `rdf:_i` predicates are considered the same property, then container membership properties are the 6th most common predicate after `rdf:type`, `dbpedia:wikilink`, `rdfs:seeAlso`, `foaf:knows`, and `foaf:nick`. In fact, out of 136,188 unique predicates in the dataset, `rdf:_1` is the 91st most commonly occurring predicate showing up in 757,506 triples. This suggests that container membership properties enjoy significant usage.<sup>2</sup>

Having suggested that container membership properties are widely (enough) used, the question remains as to whether their entailments are useful. I believe they are. RDFS entailment infers triples of the form  $(C \text{ rdfs:member } M)$  from triples like  $(C \text{ rdf:}_i M)$ . Such inferences allow us to ask the question “does  $M$  belong to  $C$ ?” without asking “is  $M$  the first item in  $C$ ? the second? the third?” and so forth, or asking “is  $C$  a container that is related to  $M$  in a way that looks like `rdf:_i`?”.

## 4 Previous Approaches

### 4.1 Muñoz *et al.*'s *pdf Fragment*

In my experience, many systems simply choose to ignore entailments of container membership properties. Muñoz *et al.* [5, 6] propose the *pdf fragment*, a simple fragment of RDFS that excludes (among other things) container membership properties. However, in maintaining a high level of backward compatibility with the current standard of RDF(S), appropriate handling of container membership properties is still a concern.

### 4.2 Ter Horst's *Partial D\* Closure*

Ter Horst [2, 3] discusses this very issue, and as a solution, he defines the *partial D\* closure*. The beginning of the process for computing the *partial D\* closure* is stated as follows:

“Suppose that  $G$  is a generalized RDF graph and  $D$  a datatype map. Suppose that  $K$  is a nonempty subset of the positive integers  $\{1, 2, \dots\}$  chosen in such a way that for `rdf:_i`  $\in V(G)$  we have  $i \in K$ . The *partial D\* closure*  $G_{s,K}$  of  $G$  is defined in the following way. In the first step, all RDF and RDFS axiomatic triples and  $D$ -axiomatic triples are added to  $G$ , **except for the axiomatic triples including `rdf:_i` such that  $i \notin K$ .**”  
[3]

In other words, include only those axiomatic triples about `rdf:_i` resources for which the `rdf:_i` resources are actually mentioned in the original graph ( $G$ ).

### 4.3 Exhaustive Scan

Jena2's RDFS Reasoner [7] (using the “full compliance level”) essentially follows ter Horst's recommendation by first determining which `rdf:_i` resources are in the original graph and then adding their respective axiomatic triples. This is considered a preprocessing step in which a full scan is performed over the graph.

<sup>2</sup> *other interesting statistics*: `cpan.org`, `craigslist.org`, and `musicbrainz.org` are the top three domains with the most container membership properties. `rdf:Seq` is the most commonly occurring container and is the 15th most commonly occurring instance type.

“The identification of which container membership properties (properties like `rdf:_1`) are present is implemented using a preprocessing hook. The rest of the RDFS operations are implemented by explicit rule sets executed by the general hybrid rule reasoner.” [7]

Ianni *et al.* [8] provide entailment rules in their answer set programming system that also result in a complete scan of the triples.

## 5 Proposed Solution

Similar to the previously defined *finite RDFS closure* in [9], I propose a solution that complies with ter Horst’s suggestion. Simply define a set of additional entailment rules which generates only the axiomatic triples for the `rdf:_i` resources that are actually used.

```

CMPa:  ?n rdf:type rdfs:Resource ∧ [?n has the form rdf:_i where i ∈ {1,2,...}]
        → ?n rdf:type rdf:Property

CMPb:  ?n rdf:type rdfs:Resource ∧ [?n has the form rdf:_i where i ∈ {1,2,...}]
        → ?n rdf:type rdfs:ContainerMembershipProperty

CMPc:  ?n rdf:type rdfs:Resource ∧ [?n has the form rdf:_i where i ∈ {1,2,...}]
        → ?n rdfs:domain rdfs:Resource

CMPd:  ?n rdf:type rdfs:Resource ∧ [?n has the form rdf:_i where i ∈ {1,2,...}]
        → ?n rdfs:range rdfs:Resource

```

Note that all of the CMP rules have the same body with only one triple pattern. The triple pattern (`?n rdf:type rdfs:Resource`) is sufficient for finding all occurrences of `rdf:_i` resources by the following reasoning. If an `rdf:_i` resource occurs in the original graph, then it occurs in at least one of the subject, predicate, or object positions of at least one triple. If it occurs in a subject position, then rule *rdfs4a*<sup>3</sup> generates (`rdf:_i rdf:type rdfs:Resource`). If it occurs in an object position, then rule *rdfs4b*<sup>4</sup> generates the same result. If it occurs in a predicate position, then rule *rdf1*<sup>5</sup> generates (`rdf:_i rdf:type rdf:Property`), and then that triple satisfies rule *rdfs4a* to produce (`rdf:_i rdf:type rdfs:Resource`).

This approach differs from previous approaches in that producing the axiomatic triples for `rdf:_i` resources may be performed more selectively as part of rule-based reasoning rather than as an exhaustive scan of all the triples. In other words, only subjects of triples matching (`?n rdf:type rdfs:Resource`) must be inspected for `rdf:_i` resources instead of all positions of all triples (`?n1 ?n2 ?n3`).

## 6 Conclusion

Container membership properties are the predicates of nearly 2.5% of the triples in the 2009 Billion Triples Challenge dataset suggesting that they enjoy common usage, and I argue that container membership property entailments (namely `rdfs:member`) are useful. While many systems ignore container membership properties, I recommend following ter Horst’s suggestion of

<sup>3</sup> `u a v → u rdf:type rdfs:Resource`

<sup>4</sup> `u a v → v rdf:type rdfs:Resource`

<sup>5</sup> `u a y → a rdf:type rdf:Property`

including only the axiomatic triples about `rdf_i` resources for those container membership properties that occur in the original graph. Whereas previous systems following this suggestion have implemented an exhaustive scan of the triples, I recommend four rules that can be used during rule-based reasoning to more efficiently produce said axiomatic triples and which can be used to effectively redefine the RDFS closure to make it decidable while maximally backward-compatible.

## References

1. Hayes, P.: Rdf semantics. W3C Recommendation: <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>, last accessed April 5, 2010
2. ter Horst, H.J.: Extending the rdfs entailment lemma. In: Proceedings of the Third International Semantic Web Conference. (2004)
3. ter Horst, H.J.: Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Journal of Web Semantics* (2005)
4. Mika, P.: Social Networks and the Semantic Web. PhD thesis, Vrije Universiteit (2006)
5. Munoz, S., Perez, J., Gutierrez, C.: Minimal Deductive Systems for RDF. In: Proceedings of the 4th European Semantic Web Conference. (2007)
6. Munoz, S., Perez, J., Gutierrez, C.: Simple and Efficient Minimal RDFS. *Journal of Web Semantics* (2009) 220–234
7. Reynolds, D.: Jena 2 inference support. <http://jena.sourceforge.net/inference/>, last accessed April 5, 2010
8. Ianni, G., Martello, A., Panetta, C., Terracina, G.: Efficiently Querying RDF(S) Ontologies with Answer Set Programming. *Journal of Logic and Computation* **19**(4) (2008) 671–695
9. Weaver, J., Hendler, J.A.: Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples. In: Proceedings of the 8th International Semantic Web Conference. (2009) 682–697