Finite Models in RDF(S), with datatypes

Peter F. Patel-Schneider[1] and Pat Hayes[2]

[1] Nuance Communications, pfpschneider@gmail.com
[2] IHMC, phayes@ihmc.us

The details of reasoning in RDF [2] and RDFS [1] are generally well known. There is a model-theoretic semantics for RDF [3, 4] and there are sound and complete proof theories for RDF without datatypes [6]. However, the model-theoretic characteristics of RDF[3] have been less studied, particularly when datatypes are added. We show that RDF reasoning can be performed by only considering finite models or pre-models, and sometimes only very small models need be considered.

Ter Horst [6] does define Herbrand models for RDF and RDFS, providing the basis for some model-theoretic characteristics of RDF and RDFS, but he does not provide a full analysis of RDF datatypes, analyzing instead an incomplete semantics for datatypes that is easier to reason in. As well, the recent minor modifications to the semantics of RDF [4], while cleaning up some aspects of entailment in RDF, do make some technical changes that might appear to interfere with finite model-based reasoning in RDF. An analysis of finite models for RDF shows that the modified semantics does not introduce any unintended changes to reasoning in RDF. As well, it provides insights into the modeling strength of RDF, particularly when blank nodes are not present, and illustrates how finite datatypes interact with the rest of RDF.

As shown by ter Horst, sound and complete reasoning in RDF and RDFS without datatypes is decidable, even though RDF and RDFS have an infinite number of axioms. However, the decidability of RDF reasoning does not necessarily mean that RDF reasoning can be done by considering only finite models. For example, OWL [5] has decidable reasoning, but nonetheless requires infinite models, for example by encoding the number line using inverse properties and number restrictions.

In the new semantics for RDF [4], all IRIs are given denotations in all interpretations. This means that the standard construction for Herbrand models will result in an infinite model. Infinite datatypes (e.g., xsd:integer) also produce infinite models, so finite model reasoning in RDF with datatypes is technically concerned with pre-models, semantic structures that are finite and can be trivially extended to real models. Nonetheless finite model reasoning should be possible in RDF as RDF does not have inverses, counting, or even equality and inequality.

Given an RDF graph (or set of RDF graphs), we define the set of *identifiers* for the graph as the nodes and predicates of the graph plus the IRIs used in the RDF (and RDFS, if considering RDFS entailment) semantic conditions and

[3] In this paper, RDF by itself will generally be used to indicate both RDF and RDFS, both with and without datatypes, unless otherwise specified.

axioms, except that no container membership property not occurring in the graph is an identifier for the graph.

We then build some models for the RDF graph as follows.[4] We start with the data values for the recognized datatypes. For every identifier that is not a literal with a recognized datatype we nondeterministically either nondeterministically choose some data value as its denotation or add a new domain element as its denotation. This results in denotation functions where identifiers that do not denote data values all denote different domain elements. For IRIs and literals with unrecognized datatypes that are not identifiers of the graph we add two extra domain elements, one being the denotation of the container membership properties that are not identifiers for the graph and one being the denotation of all other identifiers. We then build up the rest of the semantic structure using the graph and the axioms and rules of inference from ter Horst augmented with axioms for datatypes and co-denoting identifiers, resulting in a structure like a datatype-aware Herbrand interpretation except that some non-literal identifiers might denote data values.

Some of these denotation functions might fail to produce an interpretation because some datatype domain or range restriction requires a domain element to be a data value for a particular datatype when it is not. However, if there is a model for the RDF graph, then this construction will produce at least one model, because there are no semantic conditions in RDF that require a particular denotation for an identifier or require or prohibit co-denotation between identifiers except those related to data values and we have not constrained data value denotations here except for non-identifiers.

So for every satisfiable RDF graph we have ended up with a set of models. We now need to show that any model of the graph is at least as strong as one of these models. For a particular set of denotations the inference rules produce the weakest possible model. Now consider the extra domain element added for unmentioned container membership properties. Replicating this domain element and splitting denotations produces a model that has the same strength as the original model because the RDF semantic conditions treat all these domain elements the same and identity cannot be detected. Similarly replicating the other additional domain element produces a model of the same strength. Because there is no inequality in RDF, identifying any two domain elements always produces model that is at least as strong, if it produces a model at all. Thus the restriction that denotations that are not data values be unique produces the weakest possible models.

In these models all the data values in a datatype that are not the denotation of some identifier have exactly the same characteristics. A pre-model can thus be constructed that collapses all these data values into one, finally resulting in finite model reasoning for RDF. (The result is, of course, not generally a model because it violates the semantic conditions on the denotation of literals.) A completely finite semantic structure can be constructed by simply ignoring these denotation mappings and the mappings for other non-identifiers.

[4] For purposes of space some shortcuts in notation will be taken throughout this paper.

In the absence of recognized datatypes, the above construction results in unique Herbrand models just like the ones in ter Horst. In the presence of recognized datatypes this construction is different from that in ter Horst, as it captures the full meaning of datatypes, including the requirement to consider several models. For example, consider a datatype with only two data values, say ex:two, and the RDF graph

ex:p rdf:range ex:two.
ex:a ex:p ex:u, ex:v. ex:b ex:p ex:u, ex:w. ex:c ex:p ex:v, ex:w.

This RDF graph entails _:x ex:p ex:u, ex:v, ex:w. To determine entailment in these situations more than one model must be considered, hence the choice of values in the graph above.

If datatypes have sufficient data values of the right kind, however, then it is possible to only consider models that are more like Herbrand models. Given a finite set of recognized datatypes D and E a subset of D, let the unconstrained portion of E be the elements of the intersection of the data spaces for each e in E that are not in any other datatype in E that is not a superset of the intersection. Consider two RDF graphs A and B and a set of recognized datatypes D. If the unconstrained portion of every E, a non-empty subset of D, is of size greater than the number of data values in it denoted by literals in A and B plus the number of identifiers in A that are not literals with recognized datatypes plus one then it is possible to always choose unconstrained elements when picking data values for identifiers that are not literals with recognized datatypes. Then all such identifiers will have different denotations, and different denotations from all literals. This in turn permits the determination of the datatypes that the denotation of an identifier must belong to by using the D* rules of ter Horst. Then when determining the denotation of an identifier, if this set is empty add a new domain element and otherwise pick an unconstrained value for this set. This results in a single, finite model that can be used for reasoning. Note, however, that the presence of even a single too-small unconstrained portion may require examining multiple models.

So we have shown that RDF reasoning can be done by considering only models of the size of the RDF graph. Is it possible to consider only very small models? (Datatypes make these considerations even more complex, so this section of the paper will ignore datatypes.) If RDF had disjunction, then it would not be possible to significantly shrink the minimum size of considered models. For example, consider RDF graphs containing n triples of the form

S_i S_1 S_i. for $1 <= i <= n$.

In any model with less than n domain elements, there is some $1 \leq i \neq j \leq n$ such that S_i and S_j have the same denotation. In this model S_i S_1 S_j. is true and so in any such model the disjunction of all these triples is true, which is not a valid entailment.

Even with RDF lacking disjunctions, it is possible to show that very small models are not adequate. Consider RDF graphs containing triples of the form

S_i S_1 S_j . for $1 \leq i \neq j \leq n$.

In any model with less than n domain elements, there is some $1 \leq i \neq j \leq n$ such that S_i and S_j have the same denotation, which is then related to the denotation of each of the S_i by the denotation of S_1, so the graph containing

_:x S_1 S_j. for 1 <= i <= n,

is true in each of these models, but this graph is not entailed. Therefore considering only models of this size or smaller is not sufficient.

If we only consider entailments with no blank nodes in the entailed graph then smaller models suffice. Consider an interpretation I (for RDF or RDFS without any recognized datatypes) containing two domain elements e_1 and e_2 that are neither properties nor classes (call these domain elements *ordinary*). Form I' from I by simply replacing e_1 and e_2 with a single domain element e throughout. Then I' is an interpretation, which can be determined by examining all the appropriate semantic conditions.

So for B_1 and B_1 identifiers whose denotation in I are neither e_1 nor e_2, I' supports any triple of the form B_1 P B_2., if and only if I supports the triple. For any particular such triple this process can be repeated until only three ordinary domain elements remain. Considering all such shrunken interpretations is adequate to rule out any invalid entailments, so we need only consider models with three ordinary domain elements, but of course we need to consider many interpretations. The ability to have such small models and to then only consider them shows how weak RDF is as a logic.

We have argued that RDF reasoning can be done by only considering finite models, even in the presence of datatypes. The exact size and number of the models that need to be considered depends on a number of factors, including which recognized datatypes are involved, but generally models of size at least the number of identifiers in the graph must be considered. If there are no datatypes or the datatypes are of sufficient size, then a single Herbrand-like model is all that need be considered. If there are no blank nodes in the entailed graph, then much smaller models suffice, although multiple models must then be considered.

References

1. Dan Brinkley and R. V. Guha. RDF vocabulary description language 1.0: RDF schema. W3C Recommendation, http://www.w3.org/TR/rdf-schema, 2004.
2. Richard Cyganiak and David Wood. RDF 1.1 concepts and abstract syntax. W3C Working Draft, http://www.w3.org/TR/rdf11-concepts, 2013.
3. Patrick Hayes. RDF Semantics. W3C Recommendation, http://www.w3.org/TR/rdf-mt/, 2004.
4. Patrick Hayes and Peter F. Patel-Schneider. RDF 1.1 Semantics. W3C Working Draft, http://www.w3.org/TR/rdf11-mt/, 2013.
5. Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 web ontology language: Structural specification and functional-style syntax. W3C Recommendation, http://www.w3.org/TR/owl2-syntax/, 2009.
6. Herman J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2-3):79–115, 2005.