Model Theory and Entailment Rules for RDF Containers, Collections and Reification

François Bry[1] and Tim Furche[1] and Benedikt Linse[1]

University of Munich,
http://www.pms.ifi.lmu.de

Abstract. An RDF graph is, at its core, just a set of statements consisting of subjects, predicates and objects. Nevertheless, since its inception practitioners have asked for richer data structures such as containers (for open lists, sets and bags), collections (for closed lists) and reification (for quoting and provenance). Though this desire has been addressed in the RDF primer and RDF Schema specification, they are explicitely ignored in its model theory. In this paper we formalize the intuitive semantics (as suggested by the RDF primer, the RDF Schema and RDF semantics specifications) of these compound data structures by two orthogonal extensions of the RDFS model theory (RDFCC for RDF containers and collections, and RDFR for RDF reification). Second, we give a set of entailment rules that is sound and complete for the RDFCC and RDFR model theories. We show that complexity of RDFCC and RDFR entailment remains the same as that of simple RDF entailment.

1 Introduction

Being endorsed by the W3C and equipped with a model theoretic semantics, RDF/S[1] is the data format destined to form the basis of the Semantic Web. A large number of RDFS vocabularies, such as FOAF, Dublin Core, DOAP, RSS, SKOS, XMP, etc. have been proposed, and are used by popular software such as Firefox, Thunderbird, Photoshop and Acrobat. Due to its simplicity and precise semantics, RDF/S is amenable for computer processing, reasoning and understanding. Path, query and rule languages such as SPARQL [1], RQL [2], Versa [3], nSPARQL [4], RDFLog [5, 6] and XCERPT$^{\text{RDF}}$ [7] can be used to process RDF data.

There is, however, a gap within the specification of the semantics of RDF/S. While the largest part of the vocabulary is reflected in the model theory, the vocabulary employed for describing RDF containers, collections and reification is barely reflected in the model theory, although their intuitive semantics is mostly well-understood and informally specified in [8], [9] and [10].

Containers and Collections are the recommended way for describing groups of things within RDF. Containers may either be *bags* for describing multi-sets

[1] We use RDF/S to refer to RDF data considered either under the RDF or the RDFS semantics.

of things, *alternatives* for describing choices among multiple items or *sequences* for describing groups of resources for which the order is relevant. Containers are meant to be *open* in that there may be other elements than the ones in a given RDF graph. In contrast, RDF collections are meant to be *closed.* While one could imagine many types of collections, only one type of collection, namely RDF lists, has been specified by the W3C. The RDFCC model theory presented in this contribution formalizes the intuitions of *openness*, *closedness* and *order.*

RDF reification is used to describe information about statements. It can be used to express beliefs, assumptions, or fears, as well as meta data about statements, such as their originators, their time and place. Neither do statements imply their reifications nor vice versa.

The RDF primer explains the absence of a model theory for the RDF specificities as follows:

> "It is important to understand that while these types of containers are described using predefined RDF types and properties, any special meanings associated with these containers, e.g., that the members of an Alt container are alternative values, are only intended meanings. These specific container types, and their definitions, are provided with the aim of establishing a shared convention among those who need to describe groups of things. All RDF does is provide the types and properties that can be used to construct the RDF graphs to describe each type of container. RDF has no more built-in understanding of what a resource of type `rdf:Bag` is than it has of what a resource of type `ex:Tent` (discussed in Section 3.2) is. In each case, applications must be written to behave according to the particular meaning involved for each type."[2] [8]

Providing an "intended meaning" without a formal semantics gives rise to confusion among authors of RDF documents on the one hand and users and creators of RDF query and transformation languages on the other.

> "This concept of promoting intended semantics without formalization – because there is no formal entailment associated with the semantics of containers – will most likely continue to generate some confusion in the future about exactly what is meant when one uses a specific container." [11]

In fact,the lack of a formal semantics for containers, collections and reification has already impacted their adoption. Even W3C recommendations such as SPARQL shun specific support for containers, collections and reification beyond mere syntax. Thus, there is a danger that different RDF communities invent different, potentially incompatible replacements for these concepts.

The RDF Semantics document [9][Section 3.3] anticipates the development of *semantic extensions* for RDF/S, and mentions some intuitive consequences

[2] While this quote addresses the use of containers (i.e. RDF bags, alternatives and sequences) only, the RDF Primer takes the same stance with respect to RDF collections and reification.

for RDF collections, containers and reification, which are not formalized by the model theory, and which are thus candidates for a semantic extension. This paper provides such an extension, and tries to mirror the intuitive semantics as close as possible. We thereby prepare the ground for a consistent treatment and understanding of RDF containers, collections and reifications in all RDF documents, tools and query languages.

Containers, collections and reifications and their semantics have been studied before. [12] proposes a concise nested representation of RDF containers, collections and reifications, and provides transformations from this nested representation to RDF graphs and conversely. In contrast to [12] we do not introduce any new syntax, but adhere to the approach of *semantic extensions* as suggested by [9].

[13] proposes a semantics for RDF reification (but not for containers and collections) based on F-Logic, but does not provide a model theoretic semantics or entailment rules. Our approach is different from [13] in that we provide two kinds of semantics for RDF reification: a cautious semantics which allows different reifications of the same RDF statement, and a brave semantics that assumes that the identity of a statement is given by its subject, predicate and object. The approach in [13] corresponds to the brave reification semantics introduced in this paper.

Section 2 recapitulates the basic notions of RDF triples, graphs, RDF/S interpretations, RDF/S entailment and RDF/S entailment rules. The remainder of this paper is aligned along its contributions:

- Section 3 reconsiders the incompleteness-problem of the RDFS entailment rules described in [14] and propose an alternative solution named RDFS$^+$ that does not require blank nodes in predicate position. A sound and complete set of entailment rules for RDFS$^+$ is presented, and RDFS$^+$ entailment is shown to have the same computational properties as RDFS entailment.
- Section 4 presents the RDFCC model theory for containers and collections, and a set of entailment rules that is shown to be sound and complete with respect to the RDFCC model theory. We argue that the RDFCC model theory captures the intuitive semantics of RDF containers and collections.
- Section 5 provides a model theory and sound and complete entailment rules for the reification vocabulary. Again, we argue that the RDFR model theory captures the intuitive semantics of RDF reification.
- Also in Sections 4 and 5 RDFCC and RDFR entailment are shown to have the same computational properties as RDFS entailment.
- Finally, Section 6 provides suggestions on query primitives for containers, collections and reifications to be incorporated in SPARQL, which is the predominant RDF query language as of today.

2 Preliminaries

Being intended not only for computer processing but also for *computer understanding*, RDF information is represented in simple sentences of the form

$(subject, predicate, object)$. These sentences are called RDF triples, or, alternatively, RDF statements. RDF triples and RDF graphs (sets of triples) are formed over RDF vocabularies, i.e. three disjoint fixed sets of URIs[3], Strings (called literals in RDF) and blank node identifiers. Blank nodes in RDF graphs are considered as existentially quantified variables, and give rise to an interesting field of research within the Semantic Web community. In this paper, we consider mostly ground RDF graphs (i.e. graphs without blank nodes) because the issues of blank nodes are orthogonal to the semantics of containers, collections and reifications.

Definition 1 ((ground) RDF triple, (ground) RDF graph). *Let U be a set of URIs, L a set of literals and B a set of blank node identifiers. An* RDF triple *over U, B, L is an element in $((U \cup B) \times U \times (U \cup B \cup L))$. A* ground RDF triple *over U, B, L is an element in $(U \times U \times (U \cup L))$. A (ground)* RDF graph *over U, B, L is a set of (ground) triples over U, B, L.*

As in first order logic, the semantics of RDF graphs is fixed via interpretations. RDF interpretations are somewhat unusual in that they map predicates via the function IS to elements of the domain IR, which are subsequently mapped via $IEXT$ to subsets of $IR \times IR$. This approach of *giving names to predicate relations* has the benefit that the set of relations becomes countable, while the set of subsets of $(IR \times IR)$, i.e. the set of unnamed binary relations, is, by Cantors theorem, uncountable if IR is infinite.

As a second peculiarity, RDF interpretations must deal with RDF literals. RDF literals are treated differently from URIs, because they are completely described by the sequence of characters they are made up of. On the other hand, living beings, artifacts or concepts cannot be completely described by a single string of characters, and have thus a surrogate identity given by their URI. Also the treatment of RDF literals is orthogonal to the semantics of containers, collections and reifications. Therefore we do not further discuss RDF literals in this contribution. There are (at least) five increasingly restrictive definitions of interpretations in RDF/S: simple RDF interpretations, RDF interpretations, RDFS interpretations, RDFS datatype interpretations and OWL interpretations. Here, only simple RDF interpretations and RDFS interpretations are considered, since RDFCC and RDFR interpretations are based upon them.

Definition 2 (Simple RDF interpretation [9]).

A Simple interpretation of an RDF vocabulary $V = U \cup L$ is a 6-tuple $(IR, IP, IEXT, IS, IL, LV)$ where

- $IR \neq \emptyset$*: the* domain *or* universe *of I.*
- IP*: the* set of properties *of I.*
- $IEXT : IP \to \mathcal{P}(IR \times IR)$.

[3] URIs are often abbreviated by qualified names consisting of a namespace prefix and a local part. In this contribution we omit the well-known namespace prefixes `rdf:`, `rdfs:` and `owl:`, when the namespace prefix is clear from the context.

- $IS : U \to IR \cup IP$.
- IL : *a mapping from typed literals in* V *into* IR.
- $LV \subseteq IR$: *the* set of literal values

Simple RDF interpretations only give the semantics of URIs and typed literals. The semantics of RDF triples, graphs and untyped literals on the other hand is fixed by RDF denotations. Put briefly, denotations give the structurally recursive application of interpretations to ground triples and graphs, thereby defining their truth values.

A ground triple (s, p, o) is true in an interpretation I, if $(I(s), I(o)) \in IEXT(I(p))$, and a ground RDF graph is true in I if all its triples are true in I. As an immediate consequence, a triple (or a graph) that contains a URI not in the vocabulary of an interpretation I is always false under I. RDF denotations may also be based on RDFS, RDFCC, RDFR, RDFD or other, more restrictive notions of RDF interpretations. See [9] or Section A.1 for the Definition of RDF denotations.

RDF interpretations extend simple RDF interpretations by further restricting the usage of the `type` predicate, and XML literals. We do not discuss RDF interpretations but refer to [9].

RDFS interpretations assign a semantics to the RDFS classes [4] `Class`, `Resource`, `Literal`, `Datatype`, `ContainerMembershipProperty`, and to the RDFS predicates `domain`, `range`, `subPropertyOf`, `subClassOf`, `member`. Besides the mappings IS for URIs, IL for typed literals and $IEXT$ for predicates, RDFS interpretations use yet another mapping $ICEXT$ that maps classes to their class extensions. See [9] or Section A.1 for the definition of RDFS interpretations.

For RDF and especially for RDFS, so-called *entailment rules* have been identified, which can be used to derive additional knowledge from RDF graphs. While the definitions of (simple) RDF/S interpretations impose constraints on the valid interpretations of a graph only, entailment rules are *syntactical* transformations on RDF graphs, such that the model relationship between interpretations and the graphs to be transformed remain untouched. For simple RDF entailment, these entailment rules are very restricted. Given an RDF graph consisting of the single triple (a, b, c), one may add other triples to the graph in which the subject, the object or both are replaced by a blank node identifier. For RDFS entailment, these transformations include entailment rules for the transitivity of the subclass relationship, the subproperty relationship, entailment rules for the special predicates `domain` and `range`. For the sake of brevity, we do not give the complete list of RDFS entailment rules, but instead refer to [9], Section 7.3. Validity of these transformation rules is defined as follows:

Definition 3 (Valid and invalid entailment rules on RDF graphs). *Any entailment* $S \longrightarrow E$ *is* valid *if* S *entails* $S \cup E$ *in every case. Otherwise it is* invalid*.*

[4] As in the rest of this contribution, the namespace prefixes `rdf:` and `rdfs:` are omitted.

Definition 3 is based on the notion of entailment. Since there are different notions of entailment in RDF/S, we speak of *simply valid entailments* for simple RDF entailment, *valid entailments* for RDF entailment, *valid RDFS entailments* for RDFS entailment, etc.

3 RDFS$^+$ Entailment

The RDFS entailment rules have been shown to be incomplete in [14] for the following reason: Given two RDF properties p_1 and p_2 with p_1 subproperty of p_2, one can derive that for a triple (a, p_1, b) also the triple (a, p_2, b) must hold (RDFS entailment rule `rdfs7`). Moreover, given a property p with domain u, and an RDF triple (v, p, w), the triple (v, type, u) is entailed (RDFS entailment rule `rdfs2`). Now consider the following RDF graph:

$$\begin{aligned} G = \{\ & (\text{friend}, \text{supPropertyOf}, \text{\_:Knows}), (\text{\_:Knows}, \text{domain}, \text{Person}), \\ & (\text{john}, \text{friend}, \text{chuck})\} \end{aligned} \tag{1}$$

The graph G in Equation 1 RDFS-entails the triple $(john, type, Person)$, but this triple cannot be derived by the entailment rules `rdfs2` and `rdfs7`, since it would require an intermediate triple with a blank node in predicate position, which is forbidden in RDF. As a solution [14] proposes the notion of *generalized RDF graphs*, which are RDF graphs that allow blank nodes in predicate position.

In this contribution, we take a different approach. We argue that the RDFS semantics is unintuitive in one respect: It does not transfer the domain and range specifications from super-properties to subproperties. To see this, consider the RDF graph H in Equation 2. It is the same as G except that for the blank node `_:Knows`, we use a URI `knows`, serving both as a resource and a predicate name. While it is clear that H RDFS-entails the triples $(john, knows, chuck)$ and $(john, type, Person)$, does H also RDFS-entail the triple $(friend, domain, Person)$?

$$\begin{aligned} H = \{\ & (\text{friend}, \text{supPropertyOf}, \text{knows}), (\text{knows}, \text{domain}, \text{Person}), \\ & (\text{john}, \text{friend}, \text{chuck})\} \end{aligned} \tag{2}$$

According to the RDFS model theory, it does not. However, it is intuitive that the domain of a subproperty p_1 is a subclass of the domain of the corresponding superproperty p_2, since the RDFS model theory requires that $IEXT(I(p_1)) \subseteq IEXT(I(p_2))$. Moreover, altering the RDFS model theory such that domain and range specifications are inherited from superproperties to subproperties, results in no additional entailments other than just this inheritance. To see this, consider the graph $RDFS(H)$ which is the closure of H under the RDFS entailment rules:[5]

[5] We omit blank nodes introduced by the RDF entailment rules.

$$RDFS(H) = \{\ (\text{friend}, \text{supPropertyOf}, \text{knows}), (\text{knows}, \text{domain}, \text{Person}),$$
$$(\text{john}, \text{friend}, \text{chuck}), (\text{john}, \text{knows}, \text{chuck}), (\text{john}, \text{type}, \text{Person})\}$$

Adding the triple $(friend, domain, Person)$ would once again allow the derivation of $(john, type, Person)$, but no other additional triples. This is easy to see, since `domain` and `range` appear as premises only in the RDFS entailment rules `rdfs2` and `rdfs3`.[6]

We thus propose to add to the Definition of RDFS interpretations the semantic condition in 4 and to the RDFS entailment rule Equations 3 and 4.

Definition 4. *An RDFS$^+$ interpretation is an RDFS interpretation with the following additional semantic conditions:*[7]

- *If* $(p_1, p_2) \in IEXT(I(subPropertyOf))$ *and* $(p_2, C) \in IEXT(I(domain))$ *then* $(p_1, C) \in IEXT(I(domain))$.
- *If* $(p_1, p_2) \in IEXT(I(subPropertyOf))$ *and* $(p_2, C) \in IEXT(I(range))$ *then* $(p_1, C) \in IEXT(I(range))$.

Analogously to the other RDF entailment relationships, we say that an RDF graph H *RDFS$^+$*-entails an RDF graph G, iff all RDFS$^+$ interpretations of H are also RDFS$^+$ interpretations of G.

$$\frac{(p_1, \text{subPropertyOf}, p_2), (p_2, \text{domain}, C)}{(p_1, \text{domain}, C)} \tag{3}$$

$$\frac{(p_1, \text{subPropertyOf}, p_2), (p_2, \text{range}, C)}{(p_1, \text{range}, C)} \tag{4}$$

We denote the union of the set of RDFS entailment rules from [9] and Equations 3 and 4 as the set of $RDFS^{(+)}$ entailment rules.

Theorem 1 (Completeness and Complexity of the RDFS$^+$ entailment). *Given two RDF graphs H and G, H RDFS$^+$-entails G, if and only if an instance of G is in the RDFS$^{(+)}$ closure of H. RDFS$^+$ entailment is NP-complete in general, but in P if H is ground.*

See Section A for the proof of Theorem 1.

4 RDFCC Model Theory

In this section, we introduce the RDFCC extension of RDFS, i,e, the formalization of the intuitive semantics of RDF containers and collections as specified in

[6] Obviously, this reasoning is based on the correctness and completeness of the RDFS entailment rules.

[7] Namespace prefixes are omitted for the sake of brevity.

[8], [9] and [10]. Moreover we give a sound and complete set of entailment rules as a purely syntactical characterization of RDFCC entailment.

The RDF container vocabulary consisting of the names `Bag`, `Seq`, `Alt`, `List`, `first`, `rest`, `nil`, `_1`, `_2`, ... must be used with care in order to respect its intuitive semantics:

Definition 5 (RDFCC interpretation). *An RDFCC interpretation is an RDFS interpretation (see Definition 12) which satisifes the following conditions:*

1. $(x, y) \in IEXT(\_i), i \in \mathbb{N} \Rightarrow x \in ICEXT(Container)$[8]
2. $(x, y) \in IEXT(member) \Rightarrow x \in ICEXT(Container)$
3. $(x, y), (x, z) \in IEXT(\_i),\ i \in \mathbb{N},\ x \in ICEXT(Seq) \Rightarrow x = z.$[9]
4. $(x, y) \in IEXT(first) \cup IEXT(rest) \Rightarrow x \in ICEXT(List).$[10]
5. $(x, y), (x, z) \in IEXT(first) \Rightarrow x = z.$
6. $(x, y), (x, z) \in IEXT(rest) \Rightarrow x = z.$
7. $x \in ICEXT(List) \Rightarrow x = I(nil) \vee$ $(\ \exists y, z \in IR\ .\ (x, y) \in IEXT(first) \wedge (x, z) \in IEXT(rest)\ ).$
8. $(x, y) \in IEXT(\_i),\ i \in \mathbb{N} \Rightarrow \exists\ y_1, \ldots y_{i-1} \in IR\ .\ (x, y_j) \in IEXT(\_j)\ \forall\ 1 \leq j \leq i.$
9. *for any bijective mapping* $\pi : \{1, \ldots, n\ \} \to \{1, \ldots, n\ \}$ *holds:* $(x, y_1) \in IEXT(\_1), \ldots, (x, y_n) \in IEXT(\_n), x \in ICEXT(Alt) \Rightarrow (x, y_{\pi(i)}) \in IEXT(\_1), \ldots, (x, y_{\pi(n)}) \in IEXT(\_n)$
10. *The class extensions of bags, lists, sequences and alternatives are disjunct:* $C, D \in \{Bag, Seq, Alt, List\} \wedge C \neq D \Rightarrow$ $ICEXT(C) \cap ICEXT(D) = \emptyset$

RDF provides both the unnumbered container membership property (short UCMP) `member` and the numbered container membership properties `_1`, `_2`, ... (NCMPs). Since the intuitive semantics of alternatives is independent of the order of its elements, the `member` property suggests itself for their specification. However, RDF bags and alternatives are mostly specified using numbered container membership properties, as in the RDF Primer. Condition 9 above ensures that an RDF alternative entails all reorderings of its elements, as Example 1 illustrates.[11]

Example 1 (Semantics of NCMPs in RDF alternatives and sequences). Condition 9 states that an RDF alternative RDFCC-entails any reordering of the alternative, such that the RDF graphs in Equations 5 and 6 entail each other.

[8] This semantic condition is stronger than the RDFS axiomatic triples stating that the domains and ranges of the numbered container membership properties (NCMPs for short) `_1`, `_2`, etc are simply `Resource`.

[9] This semantic condition is equivalent to stating that the numbered container membership properties are all functional properties (`owl:FunctionalPoperty` in OWL).

[10] This semantic condition is in line with the RDFS axiomatic triples stating that the domain of `first` and `rest` is `List`.

[11] RDF bags are discussed later on in this section.

$$\{(\text{eg:alt}, \text{type}, \text{Alt}), (\text{eg:alt}, \_1, \text{eg:a}), (\text{eg:alt}, \_2, \text{eg:b})\} \tag{5}$$

$$\{(\text{eg:alt}, \text{type}, \text{Alt}), (\text{eg:alt}, \_1, \text{eg:b}), (\text{eg:alt}, \_2, \text{eg:a})\} \tag{6}$$

There is no equivalent semantic condition for RDF sequences, since for sequences, the order *is* relevant, and thus two sequences with the same members but different order must be interpreted as two different elements in IR. Thus the RDF graph in Equation 7 does not RDFCC-entail the RDF graph in Equation 8 nor the other way around. This is reflected in the RDFCC model theory in that there are RDFCC interpretations for 7 and 8 which are not RDFCC interpretations for 8 and 7, respectively.

$$\{(\text{eg:alt}, \text{type}, \text{Seq}), (\text{eg:alt}, \_1, \text{eg:a}), (\text{eg:alt}, \_2, \text{eg:b})\} \tag{7}$$

$$\{(\text{eg:alt}, \text{type}, \text{Seq}), (\text{eg:alt}, \_1, \text{eg:b}), (\text{eg:alt}, \_2, \text{eg:a})\} \tag{8}$$

Example 2 (Openness/Closedness of RDF Containers/Collections). The RDFCC model theory formalizes the intuitive notion of openness of RDF containers and closedness of RDF collections given in [8]. To see this, consider the RDF graphs H_1, H_2, G_1 and G_2 below. While the graphs H_1 and H_2 are always compatible, G_1 and G_2 are only compatible under the assumption that eg:a denotes the same resource as eg:b. Therefore the membership of `eg:x` is fixed, once it is given an `first` and an `rest` predicate.

$$\begin{aligned}
H_1 &= \{(\text{eg:x}, \text{type}, \text{Alt}), (\text{eg:x}, \text{member}, \text{eg:a})\} \\
H_2 &= \{(\text{eg:x}, \text{member}, \text{eg:b}), (\text{eg:x}, \text{member}, \text{eg:c})\} \\
G_1 &= \{(\text{eg:x}, \text{type}, \text{List}), (\text{eg:x}, \text{first}, \text{eg:a})\} \\
G_2 &= \{(\text{eg:x}, \text{first}, \text{eg:b}), (\text{eg:x}, \text{rest}, \text{nil})\}
\end{aligned}$$

The specification of RDF bags is problematic with the currently available RDF vocabulary. On the one hand, RDF bags are unordered as RDF alternatives, and thus an RDF bag should entail any reordering of its elements. On the other hand, multiple occurrences of the same element within a bag *do* matter – the bag (or multi-set) $\{a, a\}$ is different from the bag $\{a\}$. Entailment of all reorderings would thus loose the information about the cardinality of elments in a bag.

Specificiation of RDF bags with the UCMP `member` does not solve the problem either, since the model theory is set-based. A clean solution to this dilemma would be the introduction of URIs `card_1`, `card_2` . . . , which are used to specify the cardinality of an element within an RDF bag. See Section A.3 for a model theoretic formalization of RDF bags with this extended vocabulary.

With the RDFCC model theory requiring the disjointness of the class extensions of RDF bags, sequences, alternatives and lists, there are RDF graphs that have no interpretation under the RDFCC semantics. One such graph is

$\{(a, \text{type}, \text{Bag}), (a, \text{type}, \text{Alt})\}$. We call this situation an *RDFCC clash.* In the RDFS extension for datatypes, similar cases can occur: The graph

$$\{(\_\text{:x}, \text{type}, \text{xsd:string}), (\_\text{:x}, \text{type}, \text{xsd:decimal})\}$$

has no interpretation under the datatype extension of RDFS. This situation is referred to as a *datatype clash.*

RDFCC entailment is analogous to RDF simple entailment, RDF entailment and RDFS entailment: We say that an RDF graph G_1 entails an RDF graph G_2 under the RDFCC semantics (written $G_1 \models_{RDFCC} G_2$), iff all RDFCC interpretations of G_1 are also RDFCC interpretations of G_2.

The model theoretic characterization of RDFCC implies the following triples and *syntactic* entailment rules, which are valid in the sense of Definition 3.[12]

$$(\_\text{i}, \text{domain}, \text{Container}) \quad \forall i \in \mathbb{N} \tag{9}$$

$$(\text{member}, \text{domain}, \text{Container}) \tag{10}$$

$$(\text{first}, \text{domain}, \text{Container}) \tag{11}$$

$$(\text{rest}, \text{domain}, \text{Container}) \tag{12}$$

$$\frac{(a, \text{type}, \text{List}), a \neq \text{nil}}{(a, \text{first}, \_\text{:X}), (a, \text{rest}, \_\text{:Y})} \tag{13}$$

$$\frac{(a, \_\text{i}, b)}{(a, \_\text{j}, \_\text{:}Z\_j) \quad \forall 1 \leq j < i} \tag{14}$$

$$\frac{(a, \_\text{i}, b)}{(a, \text{member}, b)} \quad \forall i \in \mathbb{N} \tag{15}$$

$$\frac{(a, \_\text{i}, b), (a, \text{type}, \text{Alt})}{(a, \_\text{j}, b)} \quad \forall 1 \leq j \leq i \tag{16}$$

Note that in Equation 13 we assume the unique name property for the URI `nil`. While in general, the unique name assumption is insensible for a distributed environment like the Web, in this particular case we can argue that anybody who uses (and thus knows) the URI `List` should also know (and thus use) the URI `nil` for referencing the empty RDF list.

For an RDF graph G, $RDFCC(G)$ is the closure of G under the RDFCC and the RDF/S entailment rules. Unfortunately, the RDFCC entailment rules are not complete in the sense that they allow to derive all graphs G' RDFCC-entailed by an RDF Graph G. In particular, $G_1 \models_{RDFCC} G_2 \Leftrightarrow RDFCC(G_1) \models_{RDFS} RDFCC(G_2)$ is *not* true for arbitrary RDF graphs G_1 and G_2 as Example 3 shows.

Example 3 (Incompleteness of the RDFCC entailment rules). Consider the RDF graphs g_1 and g_2 below. Clearly g_1 RDFCC-entails g_2, but g_2 is not in the RDFCC closure of g_1. The missing entailment rules are borrowed from OWL, and given in Definition 6.

[12] `_:X`, `_:Y`, `_:Z_1`, ... denote fresh blank node identifiers.

$$g_1 = \{(\text{eg:a}, \_1, \text{eg:b}), (\text{eg:a}, \_1, \text{eg:c}), (\text{eg:c}, \text{type}, \text{foaf:Person})\}$$
$$g_2 = \{(\text{eg:b}, \text{type}, \text{foaf:Person})\}$$

To complete the RDFCC entailment rules, the `owl:functionalProperty` and `owl:sameAs` properties are necessary for the formulation of entailment rules reflecting Condition 3 of Definition 5. These additional entailment rules are given together with the RDFCC axiomatic triples $\mathcal{A}^{CC}$ in Definition 6.

Definition 6 (*$RDFCC^{(+)}$* closure). *The set $\mathcal{A}^{CC}$ is the following set of* RDFCC axiomatic triples*:*

- (*_1*, *type*, *owl:functionalProperty*)
- (*_2*, *type*, *owl:functionalProperty*)
- *. . . ,*
- (*first*, *type*, *owl:functionalProperty*)
- (*rest*, *type*, *owl:functionalProperty*)
- (*owl:sameAs*, *type*, *owl:symmetricProperty*)

The RDFCC$^+$ entailment rules *are as follows:*

$$\frac{(p, type, owl{:}functionalProperty), (a, p, b), (a, p, c)}{(a, owl{:}sameAs, c)} \tag{17}$$

$$\frac{(a, owl{:}sameAs, b), (a, pred, obj)}{(b, pred, obj)} \tag{18}$$

$$\frac{(a, owl{:}sameAs, b), (subj, a, obj)}{(subj, b, obj)} \tag{19}$$

$$\frac{(a, owl{:}sameAs, b), (subj, pred, a)}{(subj, pred, b)} \tag{20}$$

$$\frac{(p, type, owl{:}symmetricProperty), (a, p, b)}{(b, p, a)} \tag{21}$$

With $RDFCC^{(+)}$ we refer to the union of the RDFCC entailment rules and the RDFCC$^+$ entailment rules. Given an RDF graph G, $RDFCC^{(+)}(G)$ denotes the closure of $G \cup \mathcal{A}^{CC} \cup \mathcal{A}^{RDFS}$ under the $RDFCC^{(+)}$ and RDF/S entailment rules.

Theorem 2 (Soundness and Completeness of *$RDFCC^{(+)}$*). *Given two RDF graphs G_1 and G_2 which are free of RDFCC clashes, $(G_1 \cup \mathcal{A}^{CC}) \models_{RDFCC} (G_2 \cup \mathcal{A}^{CC})$ iff $RDFCC^{(+)}(G_1) \models_{RDFS} RDFCC^{(+)}(G_1)$.*

See Section A.4 for a proof sketch of Theorem 2.

Lemma 1 (Interpolation Lemma for RDFCC). *Given two RDF graphs H and G free of RDFCC clashes, $H \cup \mathcal{A}^{CC} \models_{RDFCC} G \cup \mathcal{A}^{CC}$, iff an instance of G is a subset of the $RDFCC^{(+)}$ closure of H.*

Lemma 1 is a direct consequence of the RDFS entailment lemma and Theorem 2. It gives rise to the following complexity results about RDFCC entailment:

Theorem 3 (Complexity of RDFCC entailment). *Given two RDF graphs H and G, deciding $H \models_{RDFCC} G$ is NP-complete, and in P if G is ground.*

Proof. NP-hardness of RDFCC entailment is inherited from simple entailment [14], whose NP-hardness has been shown by a reduction from the Clique problem. That proof does not make use of any vocabulary that is further constrained in the RDFCC model theory.

Polynomial time RDFCC entailment for ground target graphs would be immediate if the RDFCC closure of an RDF graph were of polynomial size. Unfortunately, this is not the case because for the following three reasons: (i) Both the set of RDFS axiomatic triples, and the set of RDFCC axiomatic triples are infinite. (ii) Rule 14 adds an apriori unknown number of triples to the closure, and (iii) Rule 16 considers an exponential number of permutations. These issues are solved as follows: (a) Only those RDFS and RDFCC axiomatic triples for the predicates `_i` are considered in the computation of the closure that are actually relevant for the graph – i.e. the predicate must occur somewhere in the graph. (b) When syntactically deciding entailment, Rule 9 need not be applied for the computation of the closure. Instead a simple integer comparison can check if for a triple $(a, \_i, b)$ in the entailed graph, there is some triple $(a, \_j, b)$ with $i < j$ in the entailing graph. (c) We replace all numbered container membership properties (NCMP) used for RDF alternatives with the unnumbered container membership property (UCMP) `member`. Moreover, we include only the RDFS axiomatic triples for those NCMPs which are used in the entailing graph, as detailed in [14]. With these three restrictions, the size of RDFCC closure of an RDF graph G is polynomial in G. Since for ground target graphs, checking entailment reduces to checking the subset relationship, this implies the second part of Theorem 3.

As for RDFS$^{(+)}$ entailment, RDFCC entailment for non-ground graphs is in NP, since we may guess a mapping ϕ from the blank nodes in the entailed graph G to the vocabulary of the entailing graph H, and subsequently check that $\phi(G) \subseteq RDFCC^{(+)}(H)$, where $RDFCC^{(+)}(H)$ is computed with restrictions (a), (b) and (c) above.

5 RDFR Model Theory

In this section we introduce the RDFR extension of RDFS, i.e. the formalization of the intuitive semantics of RDF reification. We distinguish a *cautious reification semantics* that allows different occurrences of the same reified statement, and a *brave reification semantics*, that assumes that the identity of a reified statement is given only by the values of its `subject`, `predicate` and `object` properties. As for RDFCC, we give a sound and complete set of entailment rules for the syntactical characterization of RDFR entailment, and we show that RDFR entailment is NP complete for arbitrary graphs, and in P if the target graph is ground.

Definition 7. *A* cautious RDFR interpretation *is an RDFS interpretation satisfying the following conditions and the axiomatic triples in Definition 8.*

1. *If $(x, y), (x, z) \in IEXT(p)$ for $p \in \{subject, predicate, object\}$ then $x = z$. In other words,* **`subject`**, **`predicate`**, *and* **`object`** *are functional properties.*
2. *If $(x, y) \in IEXT(p)$ for $p \in \{subject, predicate, object\}$ then $x \in ICEXT(Statement)$.*
3. *If $x \in ICEXT(Statement)$ then $\exists s, p, o \in IR$. $(x, s) \in IEXT(subject)$, $(x, p) \in IEXT(predicate)$ and $(x, o) \in IEXT(object)$.*[13]

Definition 8. *The RDFR* axiomatic triples $\mathcal{A}^{RDFR}$ *are the following:*

1. $(subject, tpe, functionalProperty)$
2. $(predicate, tpe, functionalProperty)$
3. $(object, tpe, functionalProperty)$
4. $(subject, domain, Statement)$
5. $(predicate, domain, Statement)$
6. $(object, domain, Statement)$
7. $(sameAs, type, symmetricProperty)$

Definition 9 (Brave RDFR interpretations). *A cautious RDFR interpretation is a brave RDFR interpretation, iff it additionally satisifies the following condition:*

- *if $(x, s) \in IEXT(I(subject))$, $(y, s) \in IEXT(I(subject))$, $(x, p) \in IEXT(I(predicate))$, $(y, p) \in IEXT(I(predicate))$, $(x, o) \in IEXT(I(object))$, $(y, o) \in IEXT(I(object))$ then $x = y$.*

Cautious ($\models^{c}_{rdfr}$) and brave ($\models rdfr^{b}$) RDFR entailment are defined analogously to RDFS entailment.

Definition 10 (RDFR Entailment Rules). *The* RDFR entailment *rules include the RDFCC entailment rules 17, 18, 19 and 20. Rule 22 makes the set of RDFR entailment rules complete for $\models^{c}_{rdfr}$.*[14] *$\models^{b}_{rdfr}$ entailment additionally requires Rule 23.*

$$\frac{(a, type, Statement)}{(a, subject, \_{:}S), (a, predicate, \_{:}P), (a, object, \_O)} \tag{22}$$

$$\frac{(x, subj, s), (y, subj, s), (x, pred, p), (y, pred, p), (x, obj, o), (y, obj, o)}{(x, sameAs, y)} \tag{23}$$

The RDFR closure *$RDFR(G)$ of an RDF graph G is the closure of $(G \cup \mathcal{A}^{RDFR})$ under the RDFR and the RDF/S entailment rules.*

[13] This semantic condition is stronger than the RDFS axiomatic triples stating that the domains of `subject`, `predicate`, and `object` are `Statement`. Together with these axiomatic triples, this condition ensures that any resource with an `subject` has also an `predicate` and `object` (and vice versa).

[14] `_:S`, `_:P` and `_:O` are fresh blank node identifiers.

Theorem 4 (Soundness, Completeness and Complexity of RDFR). *Given two RDF graphs H and G, $H \models^{c}_{RDFR} G$ iff $RDFR(H) \models_{RDFS} RDFR(G)$. RDFR entailment is NP-complete for arbitrary graphs and in P, if the target graph is ground.*

See Section A.5 for a proof sketch of Theorem 4.

6 Conclusion and Outlook

This contribution formalizes the semantics of RDF grouping and reification constructs and gives rules to implement this semantics. We have shown that entailment for this extension remains in the same complexity class as simple entailment.

While many queries over containers, collections and reifications are expressible over their triple serializations in query languages such as SPARQL or RDFLog, most queries requiring the absence of triples in a container, some order among elements in a container, or queries involving elements of an RDF List, are not:

- Finding all containers that contain *only* the resources `eg:a` and `eg:b` is impossible in SPARQL. While SPARQL supports negation as failure through optional graph patterns and the `unbound` keyword, we would have to test for the absence of an infinite number of RDF triple patterns (for each predicate `_i` with $i > 0$).
- Finding all RDF sequences that contain the resources `eg:a` and `eg:b` with `eg:a` appearing before `eg:b`. Again, there is an infinite number of RDF graphs that could satisfy this query.
- Testing for the absence of a particular resource within an RDF container.
- Finding all elements of an RDF list. In SPARQL this would require the specification of a path expression with an a-priori unknown depth. In path languages such as Versa or nSPARQL, such queries *are* expressible.
- Finding all RDF bags/sequences/alternatives with exactly n members with $n \geq 0$.

This calls for a representation of RDF specificities as a compound data structure instead of as sets of triples. In Xcerpt$^{\text{RDF}}$ [7], the RDF graph $H := H_1 \cup H_2$ from Example 2 would be represented as the term

```
eg:x{ bagOf { eg:a, eg:b, eg:c } }
```

and the query q_1 = `eg:x{ bagof {{ eg:b, eg:c }} }` would match with H (because double curly braces specify that there may be more elements in the data than given in q_1), but the query q_2 = `eg:x{ bagof{ eg:b, eg:c } }` would not (because single curly braces specify that there must be exactly the two elements `eg:b` and `eg:c` given in q_2). Also the other queries above are expressible in Xcerpt$^{\text{RDF}}$ in a concise and straight-forward manner. Such query primitives could be easily incorporated into SPARQL, and also should be integrated since querying containers and collections is a central requirement for RDF query languages [15]. However, the use of such primitives makes only sense under the RDFCC semantics, i.e. under consideration of the RDFCC entailments, and under the assumption that the queried graphs respect the RDFCC model theory. This work lays the foundation for such an extension of SPARQL.

References

1. Seaborne, A., Prud'hommeaux, E.: SPARQL query language for RDF. W3C recommendation, W3C (January 2008) http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.
2. Karvounarakis, G., Magkanaraki, A., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M., Tolle, K.: RQL: A functional query language for RDF. In Gray, P.M.D., Kerschberg, L., King, P.J.H., Poulovassilis, A., eds.: The Functional Approach to Data Management: Modelling, Analyzing and Integrating Heterogeneous Data. LNCS, Springer-Verlag (2004) 435–465
3. Ogbuji, C.: Versa: Path-based RDF query language (2005)
4. Pérez, J., Arenas, M., Gutierrez, C.: nSPARQL: A navigational language for RDF. In Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T.W., Thirunarayan, K., eds.: International Semantic Web Conference. Volume 5318 of Lecture Notes in Computer Science., Springer (2008) 66–81
5. Bry, F., Furche, T., Ley, C., Linse, B., Marnette, B.: RDFLog: It's like datalog for RDF. In: Proceedings of 22nd Workshop on (Constraint) Logic Programming, Dresden (30th September–1st October 2008). (2008)
6. Bry, F., Furche, T., Ley, C., Linse, B., Marnette, B.: Taming existence in RDF querying. In Calvanese, D., Lausen, G., eds.: RR. Volume 5341 of Lecture Notes in Computer Science., Springer (2008) 236–237
7. Bry, F., Furche, T., Linse, B., Pohl, A.: XcerptRDF: A pattern-based answer to the versatile web challenge. In: Proceedings of 22nd Workshop on (Constraint) Logic Programming, Dresden, Germany (30th September–1st October 2008). (2008) 27–36
8. Manola, F., Miller, E.: RDF primer, W3C recommendation. Technical report, W3C (2004)
9. Hayes, P.: RDF semantics. World Wide Web Consortium, Recommendation REC-rdf-mt-20040210 (February 2004)
10. McBride, B.: Rdf vocabulary description language 1.0: RDF schema (2004)
11. Powers, S.: Practical RDF. O'Reilly & Associates, Inc., Sebastopol, CA, USA (2003)
12. Conen, W., Klapsing, R., Kppen, E.: RDF M&S revisited: From reification to nesting, from containers to lists, from dialect to pure xml. In: In Proceedings of the Semantic Web Working Symposium (SWWS). (2001) 2–7
13. Yang, G., Kifer, M.: Reasoning about anonymous resources and meta statements on the semantic web. J. Data Semantics **1** (2003) 69–97
14. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. J. Web Sem. **3**(2-3) (2005) 79–115
15. Haase, P., Broekstra, J., Eberhart, A., Volz, R.: A comparison of RDF query languages. (2004) 502–517

A Appendix

A.1 Denotations and RDFS Interpretations

Definition 11 (Denotation of ground RDF graphs). *Given an RDF interpretation $I = (IR, IP, IEXT, IS, IL, LV)$ over a vocabulary V, the denotation of a ground RDF graph is defined as follows:*

- *if E is a plain literal "aaa" in V then $I(E)$ =* `aaa`
- *if E is a plain literal "aaa"@ttt in V then $I(E)$ =* `<aaa,ttt>`
- *if E is a typed literal in V then $I(E)$=$IL(E)$*
- *if E is a URI reference in V then $I(E)$=$IS(E)$*
- *if E is a ground triple (s, p, o) then $I(E)$ = true if s, p, o are in V, $I(p)$ is in IP and $(I(s), I(o))$ is in $IEXT(I(p))$.*
- *if E is a ground RDF graph then $I(E)$ = false if $I(E')$ = false for some triple E' in E, otherwise $I(E)$ = true.*

Definition 12 (RDFS interpretation [9]). *An RDFS interpretation is an RDF interpretation that satisfies the following additional conditions and the RDFS axiomatic triples $\mathcal{A}^{RDFS}$ (see [9]).*

- *$x \in ICEXT(y)$ iff $(x, y) \in IEXT(I(type))$*
- *$IC = ICEXT(I(Class))$*
- *$IR = ICEXT(I(Resource))$*
- *$LV = ICEXT(I(Literal))$*
- *If (x, y) is in $IEXT(I(domain))$ and (u, v) is in $IEXT(x)$ then u is in $ICEXT(y)$.*
- *If (x, y) is in $IEXT(I(range))$ and (u, v) is in $IEXT(x)$ then v is in $ICEXT(y)$.*
- *$IEXT(I(subPropertyOf))$ is transitive and reflexive on IP.*
- *If (x, y) is in $IEXT(I(subPropertyOf))$ then x and y are in IP and $IEXT(x)$ is a subset of $IEXT(y)$.*
- *If x is in IC then $(x, I(Resource))$ is in $IEXT(I(subClassOf))$.*
- *If (x, y) is in $IEXT(I(subClassOf))$ then x and y are in IC and $ICEXT(x)$ is a subset of $ICEXT(y)$.*
- *$IEXT(I(subClassOf))$ is transitive and reflexive on IC.*
- *If x is in $ICEXT(I(ContainerMembershipProperty))$ then $(x, I(member))$ is in $IEXT(I(subPropertyOf))$.*
- *If x is in $ICEXT(I(Datatype))$ then $(x, I(Literal))$ is in $IEXT(I(subClassOf))$.*

A.2 Proof for Theorem 1

Proof. Theorem 1 is based on the completeness of the RDFS entailment rules from [9] apart from the exception identified by [14] for blank nodes in predicate position. To see that the $RDFS^+$ extension eliminates the problem of intermediate triples with blank nodes in predicate position, note that the RDFS entailment rule `rdfs7` is the only one which may derive such invalid triples. Moreover, take notice that the $RDFS^{(+)}$ closure of the graph G in Equation 1 contains the triple $(john, \text{type}, Person)$.

NP-hardness of $RDFS^{+}$ entailment is a direct consequence of NP-hardness of $RDFS$ entailment [14]. $RDFS^{+}$ entailment is in NP, since the RDFS$^{(+)}$ closure RDFS$^{(+)}(H)$ of H can be computed in polynomial time, if the axiomatic triples for the `_i` predicates are restricted to the ones occurring in H. Entailment can then be checked by guessing the right assignment ϕ of URIs in H to blank nodes of the entailed graph G, and testing the subset relationship $\phi(G) \subseteq RDFS^{(+)}(H)$ in linear time. For ground graphs, ϕ need not be guessed, but is the empty mapping.

A.3 Formalization of RDF Bags

Definition 13 gives a formalization of the intuitive semantics of RDF bags, if the cardinality of elements within a bag are specified via the predicates `card_1`, `card_2`.

Definition 13 (Formalization of RDF bags). *An RDFCC interpretation is an RDFCCBag interpretation, if the following additional semantic conditions hold:*

1. *if $(x, y) \in IEXT(card\_i)$ for some natural number $i > 0$, then $x \in ICEXT(Bag)$.*
2. *if $(x, y) \in IEXT(card\_i)$ and $(x, y) \in IEXT(card\_j)$ for two natural numbers $i, j > 0$ then $i = j$.*

Condition 1 in Definition 13 ensures that the cardinality properties are only used for the specification of RDF bags. Condition 2 ensures that the cardinality of an element within an RDF bag is uniquely determined. Since this formalization of RDF bags introduces new vocabulary and requires a redefinition of the merge of an RDF graph, it is not part of the RDFCC model theory.

A.4 Proof Sketch for Theorem 2

The correctness of Theorem 2 is substantiated by the following assignment of entailment rules to the semantic conditions in Definition 5.

- Conditions 1 and 2 are reflected by entailment rules 9 and 10, respectively.
- Condition 3 is reflected by the $RDFCC^{+}$ entailment rules 17, 18, 19 and 20 together with the axiomatic triples (_i, type, owl:functionalProperty) for $i \in \mathbb{N}, i > 0$.
- Condition 4 is reflected by the rules 11 and 12.
- Conditions 5 and 6 are reflected by the RDFCC^{+} rules 17, 18, 19 and 20, together with the axiomatic triples (first, type, owl:functionalProperty) and (rest, type, owl:functionalProperty), respectively.
- Condition 7 is reflected by rule 13.
- Condition 8 is reflected by rule 14.
- Condition 9 is reflected by rule 16.
- Condition 10 is ensured by the assumption that the graphs are free of RDFCC clashes.

A.5 Proof Sketch for Theorem 4

The validity of the RDFR entailment rules is substantiated by the following association of entailment rules and axiomatic triples to the semantic conditions of RDFR interpretations:

- Axioms 1, 2 and 3 together with the entailment rules 17, 18, 19 and 20 are valid due to Condition 1 in Definition 7.
- Axioms 4, 5 and 6 are valid due to Condition 2 of RDFR interpretations.
- Entailment rule 22 is valid due to Condition 3 in Definition 7.

The RDFR entailment rules are complete, since the semantic conditions of Definition 7 are orthogonal to each other, and because the semantic conditions on the interpretations are spelt out by the entailment rules according to the association given above.

The complexity results are proven in the same way as for RDFS$^{(+)}$.