Minimal Deductive Systems for RDF

Sergio Muñoz[1], Jorge Pérez[2,3], and Claudio Gutierrez[4]

[1] Universidad Católica de la Santísima Concepción, Chile
[2] Pontificia Universidad Católica de Chile
[3] Universidad de Talca, Chile
[4] Universidad de Chile

Abstract. This paper presents a minimalist program for RDF, by showing how one can do without several predicates and keywords of the RDF Schema vocabulary, obtaining a simpler language which preserves the original semantics. This approach is beneficial in at least two directions: (a) To have a simple abstract fragment of RDFS easy to formalize and to reason about, which captures the essence of RDFS; (b) To obtain algorithmic properties of deduction and optimizations that are relevant for particular fragments. Among our results are: the identification of a simple fragment of RDFS; the proof that it encompasses the main features of RDFS; a formal semantics and a deductive system for it; sound and complete deductive systems for their sub-fragments; and an $\mathcal{O}(n \log n)$ complexity bound for ground entailment in this fragment.

1 Introduction

The Resource Description Framework (RDF) is the W3C standard for representing information in the Web [17]. The motivation behind the development of RDF by the W3C was, as Tim Berners-Lee pointed out for the Semantic Web, to have a common and minimal language to enable to map large quantities of existing data onto it so that the data can be analyzed in ways never dreamed of by its creators [2]. If one would like to bring to reality this vision, the processing of RDF data at big scale must be viable. The very future of RDF data deployment and their use will depend critically on the complexity of processing it.

Efficient processing of any kind of data relies on a compromise between two parameters, namely, the size of the data and the expressiveness of the language describing it. As we already pointed out, in the RDF case the size of the data to be processed will be enormous, as examples like Wordnet [12], FOAF [3] and Gene Ontology [19] show. Hence, a program to make RDF processing scalable has to consider necessarily the issue of the expressiveness of RDF. Due to the well known fact that the complexity of entailment using RDF data in its full expressiveness is an untractable problem [7,8,4], such a program amounts essentially to look for fragments of RDF with good behavior w.r.t. complexity of processing. This is the broad goal of the present paper.

The full specification of RDF (that is, including RDFS vocabulary) and their fragments has not yet been studied in detail. Its description is given in [16] and its

E. Franconi, M. Kifer, and W. May (Eds.): ESWC 2007, LNCS 4519, pp. 53–67, 2007.

semantics is defined in [15]. The first observation that arises when dealing with RDFS vocabulary is the difficulty to work with it. An example of this fact is that even the rules of deduction presented in the official RDF Semantics specification are not complete [10,8]. A second empirical observation is that several parts of the RDFS vocabulary have been depreciated, and practice shows that there are others that are hardly used or not being used at all. This makes it very hard for developers to build and optimize sound implementations and algorithms, and for theoreticians to work on this specification.

In order to illustrate the above issues, let us consider two well known RDFS specifications: *WordNet* [12] and *Friend of a Friend* (FOAF) [3]. Both schemas use only a proper subset of the RDFS vocabulary. FOAF schema has no blank nodes. Additionally, there is a point about the real need of explicitly declaring classes via `rdfs:Class`: In both specifications the triples where `rdfs:Class` occurs are redundant (i.e. can be deduced from the rest of the data). Something similar happens with terms defined as properties (`rdf:Property`). Why use all the weight of the full RDFS specification in these cases? Another example where these type of issues will arise, is the SPARQL query language specification [11], which currently does not support RDFS entailment. There is wide agreement that more expressive vocabularies must be treated orthogonally to the rest of the SPARQL features. In practice, each query will use just a small fragment of the RDFS vocabulary. For reasoning and optimization purposes, it would be useful to have a sound and complete theory of each such fragment which preserves the semantics of RDFS.

Among the most important directions of a program to develop solutions to the above mentioned problems are:

- To identify a fragment which encompasses the essential features of RDF, which preserves the original semantics, be easy to formalize and can serve to prove results about its properties.
- To study in detail the semantics of different fragments of RDF, and give sound and complete deductive system for each of them.
- To study the complexity of entailment for the vocabulary in general and in these fragments in particular, and to develop algorithms for testing entailment.

As for the first point, in this paper we identify a fragment of RDFS that covers the crucial vocabulary of RDFS, prove that it preserves the original RDF semantics, and avoids vocabulary and axiomatic information that only serves to reason about the structure of the language itself and not about the data it describes. We lift this structural information into the semantics of the language, hiding them from developers and users.

Regarding the second point, we study thoroughly all fragments of the core fragment showing that they retain the original RDFS semantics. We then study the lattice of the theories induced by these fragments, developing minimal sound and complete proof systems for them. We also calculate what are the minimal sub-theories that should be considered when reasoning with restricted vocabulary.

Finally, regarding the point of complexity of entailment, there are two main aspects of RDF to consider: the built-in vocabulary and the notion of blank nodes. For the complexity of entailment considering blank nodes, good (polynomial) cases can be derived from well known databases and constraint–satisfaction results [4,9,5]. These cases consider special forms of interaction between blank nodes that are very common in practice. On this regard, we prove that there is a notion of normalized proof for RDFS entailment which permits to treat the issue of blank nodes entailment in a way orthogonal to the treatment of RDFS vocabulary. Using this notion, results for blank nodes can be composed modularly with particular results for ground RDFS fragments, that is, not considering blank nodes semantics.

For the the ground case, from a database point of view, even current known bounds seems totally impractical. For example, the naive approach would use closure, and estimates for the size of the closure are high: we show that in the fragment presented, it is quadratic. Nevertheless, this bound is still impractical from a database point of view. On these lines, we prove that entailment can be done in time $\mathcal{O}(n \log n)$ in the worst case, where n is the size of the source data.

The paper is organized as follows. Section 2 presents standard RDF and its semantics and discusses the vocabulary design to conclude with a proposal of core fragment, called ρdf. Section 3 studies the ρdf fragment. Section 4 presents the lattice of minimal fragments of ρdf and their deductive systems. Section 5 studies complexity of entailment in the ρdf fragment. Finally, Section 6 presents the conclusion.

2 RDF Semantics

Assume there are pairwise disjoint infinite sets $\mathbf{U}$ (RDF URI references), $\mathbf{B}$ (Blank nodes), and $\mathbf{L}$ (Literals). Through the paper we assume $\mathbf{U}$, $\mathbf{B}$, and $\mathbf{L}$ fixed, and for simplicity we will denote unions of these sets simply concatenating their names. A tuple $(s, p, o) \in \mathbf{UBL} \times \mathbf{U} \times \mathbf{UBL}$ is called an *RDF triple*. In this tuple, s is the *subject*, p the *predicate*, and o the *object*. Note that –following recent developments [6,11]– we are omitting the old restriction stating that literals cannot be in subject position.

Definition 1. *An* RDF graph *(or simply a graph) is a set of RDF triples. A subgraph is a subset of a graph. The* universe *of a graph G, denoted by* universe(G) *is the set of elements in* $\mathbf{UBL}$ *that occur in the triples of G. The vocabulary of G, denoted by* voc(G) *is the set* universe(G) $\cap$ $\mathbf{UL}$. *A graph is* ground *if it has no blank nodes. In general we will use uppercase letters $N, X, Y, \ldots$ to denote blank nodes.*

In what follows we will need some technical notions. A map is a function $\mu : \mathbf{UBL} \to \mathbf{UBL}$ preserving URIs and literals, i.e., $\mu(u) = u$ for all $u \in \mathbf{UL}$. Given a graph G, we define $\mu(G)$ as the set of all $(\mu(s), \mu(p), \mu(o))$ such that $(s, p, o) \in G$. We will overload the meaning of map and speak of a map μ *from* G_1 *to* G_2, and write $\mu : G_1 \to G_2$, if the map μ is such that $\mu(G_1)$ is a subgraph of G_2.

2.1 Interpretations

The normative semantics for RDF graphs given in [15], and the mathematical formalization in [10] follows standard classical treatment in logic with the notions of model, interpretation, entailment, and so on. In those works the RDFS theory is built incrementally from Simple, to RDF, to RDFS interpretations (or structures) and models for graphs. We present here a single notion of interpretation which summarizes Simple, RDF, and RDFS interpretations in one step, and which will be used later to define the semantics of our fragment.

Definition 2. *An interpretation over a vocabulary V is a tuple*

$$\mathcal{I} = (Res, Prop, Class, Ext, CExt, Lit, Int)$$

such that: (1) Res is a nonempty set of resources*, called the* domain *or* universe *of $\mathcal{I}$; (2) $Prop$ is a set of property names (not necessarily disjoint from Res); (3) $Class \subseteq Res$ is a distinguished subset of Res identifying if a resource denotes a class of resources; (4) $Ext : Prop \to 2^{Res \times Res}$, a mapping that assigns an* extension *to each property name; (5) $CExt : Class \to 2^{Res}$ a mapping that assigns a set of resources to every resource denoting a class; (6) $Lit \subseteq Res$ the set of literal values, Lit contains all plain literals in $\mathbf{L} \cap V$; (7) $Int : \mathbf{UL} \cap V \to Res \cup Prop$, the* interpretation mapping*, a mapping that assigns a resource or a property name to each element of $\mathbf{UL}$ in V, and such that Int is the identity for plain literals and assigns an element in Res to elements in $\mathbf{L}$.*

In [15,10] the notion entailment is defined using the idea of *satisfaction* of a graph under certain interpretation. Intuitively a ground triple (s, p, o) in an RDF graph G will be true under the interpretation $\mathcal{I}$ if p is *interpreted* as a property name, s and o are *interpreted* as resources, and the interpretation of the pair (s, o) belongs to the extension of the property assigned to p.

In RDF, blank nodes work as existential variables. Intuitively the triple (X, p, o) with $X \in \mathbf{B}$ would be true under $\mathcal{I}$ if there exists a resource s such that (s, p, o) is true under $\mathcal{I}$. When interpreting blank nodes, an arbitrary resource can be chosen, taking into account that the same blank node must always be interpreted as the same resource. To formally deal with blank nodes, extensions of the interpretation map Int are used in the following way. Let $A : \mathbf{B} \to Res$ be a function from blank nodes to resources; we denote Int_A the extension of Int to domain $\mathbf{B}$ defined by $Int_A(X) = A(X)$ when $X \in \mathbf{B}$. The function A captures the idea of existentiality.

The formal definition of model and entailment for RDFS in [15,10] relies on a set of *semantics restrictions* imposed to interpretations in order to model the vocabulary, and the *a priori* satisfaction of a set of *axiomatic triples*. We refer the reader to Appendix A for a complete formal definition of the semantics of RDFS using the notion of interpretation defined here.

2.2 RDFS Vocabulary

The RDF specification includes a set of reserved words, the RDFS vocabulary (RDF Schema [16]) designed to describe relationships between resources as well

as to describe properties like attributes of resources (traditional attribute-value pairs). Table 1 (Appendix A) shows the full RDFS vocabulary as it appears in [15], and (in brackets) the shortcuts that we will use in this paper. This vocabulary has a special interpretation (see Definition 6 in Appendix A).

Roughly speaking, this vocabulary can be divided conceptually in the following groups:

(a) a set of properties rdfs:subPropertyOf [`sp`], rdfs:subClassOf [`sc`], rdfs:domain [`dom`], rdfs:range [`range`] and rdf:type [`type`].
(b) a set of classes, rdfs: Resource, rdfs:Class, rdf:Property, rdf:XMLLiteral, rdfs:Literal, rdfs:Datatype.
(c) Other functionalities, like a system of classes and properties to describe lists: rdfs:Container, rdfs:ContainerMembershipProperty, rdfs:member, rdf:List, rdf:Alt, rdf:Bag, rdf:Seq, rdf:first, rdf:rest, rdf:nil, rdf:_1, rdf:_2, ..., and a systems for doing reification: a class rdf:Statement together with properties rdf:subject, rdf:predicate, rdf:object.
(d) Utility vocabulary, like rdfs:seeAlso, rdfs:isDefinedBy, rdfs:comment, rdf:value, rdfs:label.

The groups in (b), (c) and (d) have a very light semantics, essentially describing its internal function in the ontological design of the system of classes of RDFS. Their semantics is defined by "axiomatic triples" [15]which are relationships among these reserved words. Note that all axiomatic triples are "structural", in the sense that do not refer to external data, but talk about themselves. Much of this semantics correspond to what in standard languages is captured via typing. From a theoretical and practical point of view it is inconvenient to expose it to users of the language because it makes the language more difficult to understand and use, and for the criteria of simplicity in the design of the language.

On the contrary, the group (a) is formed by predicates whose intended meaning is non-trivial and is designed to relate individual pieces of data external to the vocabulary of the language. Their semantics is defined by rules which involve variables (to be instantiated by real data). For example, rdfs:subClassOf[`sc`] is a binary property reflexive and transitive; when combined with rdf:type[`type`] specify that the type of an individual (a class) can be lifted to that of a superclass. This group (a) forms the core of the RDF language developers use, as practice is showing.

For all the above considerations, it is that group (a) forms a natural fragment of RDFS to be studied in depth. Section 3 is devoted to study this fragment, and our results will show that there are theoretical reasons that support the convenience of this choice.

3 The ρdf Fragment of RDFS

Define ρdf (read rho-df, the ρ from *restricted* rdf) to be the following subset of the RDFS vocabulary:

$$\rho\text{df} = \{\texttt{sp}, \texttt{sc}, \texttt{type}, \texttt{dom}, \texttt{range}\}.$$

Definition 3. *Let G be a graph over* ρdf. *An interpretation $\mathcal{I}$ is a model of G under* ρdf, *denoted $\mathcal{I} \models_{\rho\text{df}} G$, iff $\mathcal{I}$ is an interpretation over* ρdf $\cup$ universe(G) *that satisfies the following conditions:*

1. *Simple:*
   (a) *there exists a function $A : \mathbf{B} \to Res$ such that for each $(s, p, o) \in G$, $Int(p) \in Prop$ and $(Int_A(s), Int_A(o)) \in Ext(Int(p))$, where Int_A is the extension of Int using A.*
2. *Subproperty:*
   (a) *$Ext(Int(\texttt{sp}))$ is transitive and reflexive over $Prop$*
   (b) *if $(x, y) \in Ext(Int(\texttt{sp}))$ then $x, y \in Prop$ and $Ext(x) \subseteq Ext(y)$*
3. *Subclass:*
   (a) *$Ext(Int(\texttt{sc}))$ is transitive and reflexive over $Class$*
   (b) *if $(x, y) \in Ext(Int(\texttt{sc}))$ then $x, y \in Class$ and $CExt(x) \subseteq CExt(y)$*
4. *Typing I:*
   (a) *$x \in CExt(y) \Leftrightarrow (x, y) \in Ext(Int(\texttt{type}))$*
   (b) *if $(x, y) \in Ext(Int(\texttt{dom}))$ and $(u, v) \in Ext(x)$ then $u \in CExt(y)$*
   (c) *if $(x, y) \in Ext(Int(\texttt{range}))$ and $(u, v) \in Ext(x)$ then $v \in CExt(y)$*
5. *Typing II:*
   (a) *For each $\texttt{e} \in$ ρdf, $Int(\texttt{e}) \in Prop$.*
   (b) *if $(x, y) \in Ext(Int(\texttt{dom}))$ then $x \in Prop$ and $y \in Class$.*
   (c) *if $(x, y) \in Ext(Int(\texttt{range}))$ then $x \in Prop$ and $y \in Class$.*
   (d) *if $(x, y) \in Ext(Int(\texttt{type}))$ then $y \in Class$.*

We define G entails H under ρdf, *denoted $G \models_{\rho\text{df}} H$, iff every model under* ρdf *of G is also a model under* ρdf *of H.*

Note that in ρdf–models we do not impose the *a priori* satisfaction of any axiomatic triple. Indeed, ρdf–models does not satisfy any of the RDF/S axiomatic triples in [15,10], because all of them mention RDFS vocabulary outside ρdf. This is also the reason for the inclusion of conditions 5 in ρdf models that capture the semantics restrictions imposed syntactically by the RDF/S axiomatic triples (dom, dom, prop), (dom, range, class), (range, dom, prop), (range, range, class), and (type, range, class), and the fact that every element in ρdf must be interpreted as a property.

The next theorem shows that this definition retains the original semantics for the ρdf vocabulary:

Theorem 1. *Let $\models$ be the RDFS entailment defined in [15,10], and let G and H be RDF graphs that do not mention RDFS vocabulary outside* ρdf. *Then*

$$G \models H \text{ iff } G \models_{\rho\text{df}} H.$$

The issue of reflexivity. There are still some details to be refined in the theory of ρdf. Note that, although in ρdf–models we do not impose the *a priori* satisfaction of any triple, there are triples that are entailed by all graphs, for example the triples $(\mathtt{sp}, \mathtt{sp}, \mathtt{sp})$, $(\mathtt{sc}, \mathtt{sp}, \mathtt{sc})$, $(\mathtt{type}, \mathtt{sp}, \mathtt{type})$, $(\mathtt{dom}, \mathtt{sp}, \mathtt{dom})$, and $(\mathtt{range}, \mathtt{sp}, \mathtt{range})$. These triples are true under every ρdf model due to the fact that `sp` must be interpreted as a reflexive relation. Also, because blank nodes work as existential variables, the triples above with the subject or the object replaced by any blank node, are also true in every ρdf–model. The good news is that these are the only triples in the ρdf fragment that are satisfied by every model:

Proposition 1. *Let t be an RDF triple such that $\models_{\rho\text{df}} t$. Then, either $t \in \{(\mathtt{sp}, \mathtt{sp}, \mathtt{sp}), (\mathtt{sc}, \mathtt{sp}, \mathtt{sc}), (\mathtt{type}, \mathtt{sp}, \mathtt{type}), (\mathtt{dom}, \mathtt{sp}, \mathtt{dom}), (\mathtt{range}, \mathtt{sp}, \mathtt{range})\}$, or t is obtained from these triples replacing the subject or object by a blank node.*

This is part of a more general phenomena, namely the presence of reflexivity for `sp` and `sc`. We will show that reflexivity for `sp` and `sc` is orthogonal with the rest of the semantics.

Definition 4 (Semantics without reflexivity of `sp` and `sc`). *An interpretation $\mathcal{I}$ is a* reflexive–relaxed *model under* ρdf *of a graph G, written $\mathcal{I} \models^{\text{nrx}}_{\rho\text{df}} G$, iff $\mathcal{I}$ is a* ρdf *model that does not necessarily satisfy the restrictions stating that $Ext(Int(\mathtt{sp}))$ and $Ext(Int(\mathtt{sc}))$ are reflexive relations over Prop and Class respectively.*

Theorem 2. *Let G and H be* ρdf *graphs. Assume that H does not contain triples of the form $(x, \mathtt{sp}, x)$ nor $(x, \mathtt{sc}, x)$ for $x, y \in \mathbf{UL}$, nor triples of the form $(X, \mathtt{sp}, Y)$ nor $(X, \mathtt{sc}, Y)$ for $X \in \mathbf{B}$ or $Y \in \mathbf{B}$. Then,*

$$G \models_{\rho\text{df}} H \text{ iff } G \models^{\text{nrx}}_{\rho\text{df}} H.$$

Essentially the above theorem states that the only use of reflexive restrictions in RDFS models is the entailment of triples of the form $(x, \mathtt{sp}, x)$, $(x, \mathtt{sc}, x)$, or their existential versions replacing the subject or object by blank nodes. Another property of $\models^{\text{nrx}}_{\rho\text{df}}$ is that it does not entail axiomatic triples:

Corollary 1. *There is no triple t such that $\models^{\text{nrx}}_{\rho\text{df}} t$.*

3.1 Deductive System for ρdf Vocabulary

In what follows, we present a sound and complete deductive system for the fragment of RDF presented in the previous section. The system is arranged in groups of rules that captures the semantic conditions of models. In every rule, $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}$, and $\mathcal{Y}$ are meta-variables representing elements in **UBL**.

1. *Simple:*
   (a) $\frac{G}{G'}$ *for a map* $\mu : G' \to G$ (b) $\frac{G}{G'}$ *for* $G' \subseteq G$

2. *Subproperty:*
   (a) $\frac{(\mathcal{A},\mathtt{sp},\mathcal{B})\ \ (\mathcal{B},\mathtt{sp},\mathcal{C})}{(\mathcal{A},\mathtt{sp},\mathcal{C})}$ (b) $\frac{(\mathcal{A},\mathtt{sp},\mathcal{B})\ \ (\mathcal{X},\mathcal{A},\mathcal{Y})}{(\mathcal{X},\mathcal{B},\mathcal{Y})}$

3. *Subclass:*
   (a) $\frac{(\mathcal{A},\mathtt{sc},\mathcal{B})\ \ (\mathcal{B},\mathtt{sc},\mathcal{C})}{(\mathcal{A},\mathtt{sc},\mathcal{C})}$ (b) $\frac{(\mathcal{A},\mathtt{sc},\mathcal{B})\ \ (\mathcal{X},\mathtt{type},\mathcal{A})}{(\mathcal{X},\mathtt{type},\mathcal{B})}$

4. *Typing:*
   (a) $\frac{(\mathcal{A},\mathtt{dom},\mathcal{B})\ \ (\mathcal{X},\mathcal{A},\mathcal{Y})}{(\mathcal{X},\mathtt{type},\mathcal{B})}$ (b) $\frac{(\mathcal{A},\mathtt{range},\mathcal{B})\ \ (\mathcal{X},\mathcal{A},\mathcal{Y})}{(\mathcal{Y},\mathtt{type},\mathcal{B})}$

5. *Implicit Typing:*
   (a) $\frac{(\mathcal{A},\mathtt{dom},\mathcal{B})\ \ (\mathcal{C},\mathtt{sp},\mathcal{A})\ \ (\mathcal{X},\mathcal{C},\mathcal{Y})}{(\mathcal{X},\mathtt{type},\mathcal{B})}$ (b) $\frac{(\mathcal{A},\mathtt{range},\mathcal{B})\ \ (\mathcal{C},\mathtt{sp},\mathcal{A})\ \ (\mathcal{X},\mathcal{C},\mathcal{Y})}{(\mathcal{Y},\mathtt{type},\mathcal{B})}$

6. *Subproperty Reflexivity:*
   (a) $\frac{(\mathcal{X},\mathcal{A},\mathcal{Y})}{(\mathcal{A},\mathtt{sp},\mathcal{A})}$ (c) $\frac{}{(p,\mathtt{sp},p)}$ *for* $p \in \rho\mathrm{df}$
   (b) $\frac{(\mathcal{A},\mathtt{sp},\mathcal{B})}{(\mathcal{A},\mathtt{sp},\mathcal{A})\ \ (\mathcal{B},\mathtt{sp},\mathcal{B})}$ (d) $\frac{(\mathcal{A},p,\mathcal{X})}{(\mathcal{A},\mathtt{sp},\mathcal{A})}$ *for* $p \in \{\mathtt{dom},\mathtt{range}\}$

7. *Subclass Reflexivity:*
   (a) $\frac{(\mathcal{A},\mathtt{sc},\mathcal{B})}{(\mathcal{A},\mathtt{sc},\mathcal{A})\ \ (\mathcal{B},\mathtt{sc},\mathcal{B})}$ (b) $\frac{(\mathcal{X},p,\mathcal{A})}{(\mathcal{A},\mathtt{sc},\mathcal{A})}$ *for* $p \in \{\mathtt{dom},\mathtt{range},\mathtt{type}\}$

Note 1 (On rules (5a) and (5b)). As noted in [10,8], the set of rules presented in [15] is not complete for RDFS entailment. The problem is produced when a blank node X is implicitly used as standing for a property in triples like $(a, \mathtt{sp}, X)$, $(X, \mathtt{dom}, b)$, or $(X, \mathtt{range}, c)$. Here we solve the problem following the elegant solution proposed by Marin [10] adding just two new rules of implicit typing (rules 5 above).

An instantiation of a rule is a uniform replacement of the metavariables occurring in the triples of the rule by elements of **UBL**, such that all the triples obtained after the replacement are well formed RDF triples.

Definition 5 (Proof). *Let G and H be graphs. Define $G \vdash_{\rho\mathrm{df}} H$ iff there exists a sequence of graphs $P_1, P_2, \ldots, P_k$, with $P_1 = G$ and $P_k = H$, and for each j $(2 \leq j \leq k)$ one of the following cases hold:*

- *there exists a map $\mu : P_j \to P_{j-1}$ (rule (1a)),*
- *$P_j \subseteq P_{j-1}$ (rule (1b)),*

– *there is an instantiation* $\frac{R}{R'}$ *of one of the rules (2)–(7), such that* $R \subseteq P_{j-1}$ *and* $P_j = P_{j-1} \cup R'$.

The sequence of rules used at each step (plus its instantiation or map), is called a proof *of* H *from* G.

Theorem 3 (Soundness and completeness). *The proof system* $\vdash_{\rho\text{df}}$ *is sound and complete for* $\models_{\rho\text{df}}$*, that is, given graphs* G *and* H *we have*

$$G \vdash_{\rho\text{df}} H \;\; \textit{iff} \;\; G \models_{\rho\text{df}} H.$$

Corollary 2. *Define the proof system* $\vdash^{\text{nrx}}_{\rho\text{df}}$ *as* $\vdash_{\rho\text{df}}$ *by droping rules of reflexivity (rules (6) and (7)). Then for graphs* G *and* H*,*

$$G \vdash^{\text{nrx}}_{\rho\text{df}} H \;\; \textit{iff} \;\; G \models^{\text{nrx}}_{\rho\text{df}} H.$$

4 Deductive Systems for Minimal Fragments of ρdf

We will assume in the rest of the paper that the user does not redefine or enrich the semantics of the ρdf-vocabulary. In syntactical terms this means that there is no triple where this vocabulary occurs in subject or object positions. This assumption is light and can be found on almost all published RDF specifications.

To begin with, the following theorem shows that for several purposes blank nodes can be treated in an orthogonal form to ρdf vocabulary.

Theorem 4 (Normal form for proofs). *Assume* $G \vdash_{\rho\text{df}} H$*. Then there is a proof of* H *from* G *where the rule (1) is used at most once and at the end.*

Consider the lattice of fragments of ρdf in Figure 1. Given one of the fragments X, by an X-graph we will understand a graph that mention ρdf vocabulary only from X. Similarly, an X-rule is one rule (2-7) that mention ρdf vocabulary only from X.

Theorem 5. *Let* X *be one of the fragments of* ρdf *in Figure 1, and let* G *and* H *be* X*-graphs. Assume that* $G \vdash_{\rho\text{df}} H$*, then there exists a proof of* H *from* G *which only uses* X*-rules and rule (1).*

The above result is based in the observation that in a proof of H from G we can avoid the following fact: a sequence of graphs $P_i, P_{i+1}, \ldots, P_{i+j}$ produced in the proof may present vocabulary outside X, but with P_i and P_{i+j} X-graphs. This fact may impose new rules obtained from the rules of $\vdash_{\rho\text{df}}$ by a concatenation that result in a sound derivation between X-graphs. It can be shown that the only rules obtained in this way coincide actually with X-rules. A second point is that triples with vocabulary outside X, produced by the application of non X-rules are not needed and can be left out of the proof of H from G.

Theorem 5 implies that X-rules are sound and complete for $\models_{\rho\text{df}}$ in fragment X. As a direct consequence we also obtain that X-rules without considering reflexivity rules, are sound and complete for $\models^{\text{nrx}}_{\rho\text{df}}$ in fragment X.

In what follows $G|_V$ means the subgraph induced by vocabulary V, i.e. those triples having subject, or predicate, or object in V.

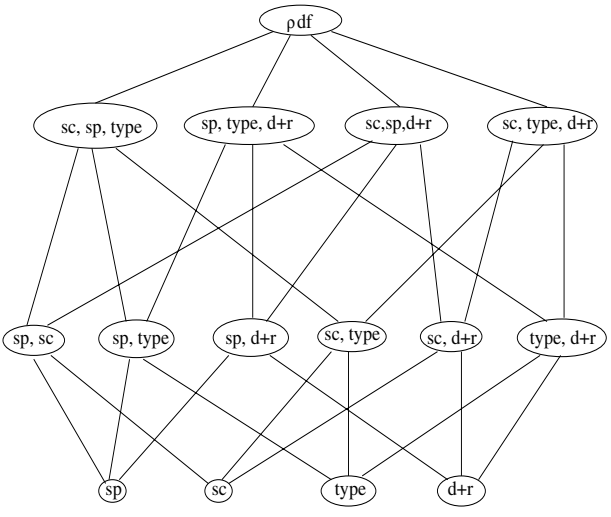

Fig. 1. The lattice of fragments of ρdf

Interpolation Lemmas for RDF. Interpolation lemmas refer to lemmas expressing the role of vocabularies in deduction. They follow from the previous results in this section.

Lemma 1. *Let G and H be graphs. If $(a, b, c) \in G$ and a, b, c do not appear in* $\mathrm{voc}(H)$ *nor in* ρdf*, then $G \models_{\rho\mathrm{df}} H$ iff $G - \{(a, b, c)\} \models_{\rho\mathrm{df}} H$.*

Lemma 2. *Let a, b, c be ground terms with b not belonging to* ρdf*. Then: $G \models_{\rho\mathrm{df}} (a, b, c)$ iff $G|_{\{\mathtt{sp},a,b,c\}} \models_{\rho\mathrm{df}} (a, b, c)$.*

Lemma 3. *Let $a, b \in$ **UBL**, then*

1. $G \models_{\rho\mathrm{df}} (a, \mathtt{dom}, b)$ *iff* $G|_{\mathtt{dom}} \models_{\rho\mathrm{df}} (a, \mathtt{dom}, b)$.
2. $G \models_{\rho\mathrm{df}} (a, \mathtt{range}, b)$ *iff* $G|_{\mathtt{range}} \models_{\rho\mathrm{df}} (a, \mathtt{range}, b)$.

Moreover, if a, b are ground, $\models_{\rho\mathrm{df}}$ reduces to membership in G.

Note 2. Although $(a, \mathtt{dom}, b)$ refers to a property a and a class b, inferring a `dom` statement in the RDFS system does not depend on statements about classes or properties. For example, from the previous lemma follows the non-intuitive fact that $\{(c_1, \mathtt{sc}, c_2), (c_2, \mathtt{sc}, c_1), (a, \mathtt{dom}, c_1)\}$ does not entail $(a, \mathtt{dom}, c_2)$.

Lemma 4. *Let $a \neq b$, then*

1. $G \models_{\rho\mathrm{df}} (a, \mathtt{sc}, b)$ *iff* $G|_{\mathtt{sc}} \models_{\rho\mathrm{df}} (a, \mathtt{sc}, b)$.
2. $G \models_{\rho\mathrm{df}} (a, \mathtt{sp}, b)$ *iff* $G|_{\mathtt{sp}} \models_{\rho\mathrm{df}} (a, \mathtt{sp}, b)$.

It turns out that `type` is the most entangled keyword in the vocabulary and deducing $G \models_{\rho\mathrm{df}} (a, \mathtt{type}, b)$ can involve all of G (except those triples mentioned in Lemma 1).

5 The Complexity of ρdf Ground Entailment

Let us introduce some notation. For a graph G and a predicate p, define G_p as the subgraph of G consisting of the triples of the form (x, p, y) of G, and define $G_\emptyset$ as the subgraph consisting of triples without ρdf vocabulary. Let $G(\mathtt{sp})$ be the directed graph whose vertices are all the elements v which appear as subject or objects in the triples of G, and in which (u, v) is an edge if and only if $(u, \mathtt{sp}, v) \in G$. Similar definition for $G(\mathtt{sc})$.

The naive approach to test the entailment $G \models H$ in the ground case would be to consider the *closure* of G and check if H is included in it. Recall that for ground G, the closure is the graph obtained by adding to G all ground triples that are derivable from G. The following result shows that this procedure would take time proportional to $|H| \cdot |G|^2$ in the worst case, which is too expensive from a database point of view.

Theorem 6. *The size of the closure of G is $\mathcal{O}(|G|^2)$, and this bound is tight.*

For the upper bound, the result follows by an analysis of the rules. The most important point is the propagation –when applicable– of the triples of the form (x, a, y) through the transitive closure of the $G(\mathtt{sp})$ graph by the usage of rule 2(b): it can be shown that this gives at most $|G_\emptyset| \times |G_{\mathtt{sp}}|$ triples. For triples having a fixed predicate in ρdf the quadratic bound is trivial. For the tightness, consider the graph $\{(a_1, \mathtt{sp}, a_2), \ldots, (a_n, \mathtt{sp}, a_{n+1})\} \cup \{(x_1, a_1, y_n), \ldots, (x_n, a_n, y_n)\}$. The number of triples of the closure of this graph is $2n + 1 + \sum_{k=1}^{n} k$ that is quadratic in n.

The following algorithm presents a much better procedure to check ground entailment in this fragment.

Algorithm (Ground Entailment)
Input: G, triple (a, p, b)

1. IF $p \in \{\mathtt{dom}, \mathtt{range}\}$ THEN check if $(a, p, b) \in G$.
2. IF $p = \mathtt{sp}$, $a \neq b$, THEN check if there is a path from a to b in $G(\mathtt{sp})$.
3. IF $p = \mathtt{sc}$, $a \neq b$, THEN check if there is a path from a to b in $G(\mathtt{sc})$.
4. IF $p \in \{\mathtt{sp}, \mathtt{sc}\}$ and $a = b$, THEN check if $(a, p, a) \in G$ else check all patterns of triples in the upper part of rules 6 (for sp) and rule 7 (for sc).
5. IF $p \notin$ ρdf THEN check $(a, p, b) \in G_\emptyset$, if it is not
   LET $G(\mathtt{sp})^*$ be the graph $G(\mathtt{sp})$ with the following marks:
   For each $(a, v, b) \in G_\emptyset$, if $v \in G(\mathtt{sp})$ then mark it green.
   IN Check in $G(\mathtt{sp})^*$ if there is a path from a vertex marked green to p
6. IF $p = \mathtt{type}$ THEN
   LET $G(\mathtt{sp})'$ be the graph $G(\mathtt{sp})$ with the following marks:
   - For each triple $(u, \mathtt{dom}, v) \in G_{\mathtt{dom}}$, if $u \in G(\mathtt{sp})$ mark the vertex u with $d(v)$.
   - For each triple $(a, e, y) \in G_\emptyset$, if $e \in G(\mathtt{sp})$, mark the vertex e with a.

LET $G(\texttt{sc})'$ be the graph $G(\texttt{sc})$ with the following marks:
- For vertex u marked $d(v)$ reachable from a vertex marked a in $G(\texttt{sp})'$, if $v \in G(\texttt{sc})$ mark it blue.
- For each $(a, \texttt{type}, w) \in G$, if $w \in G(\texttt{sc})$ mark it blue.

IN Check in $G(\texttt{sc})'$ if there is a path from a blue node to b.
Repeat this point for `range` instead of `dom`.

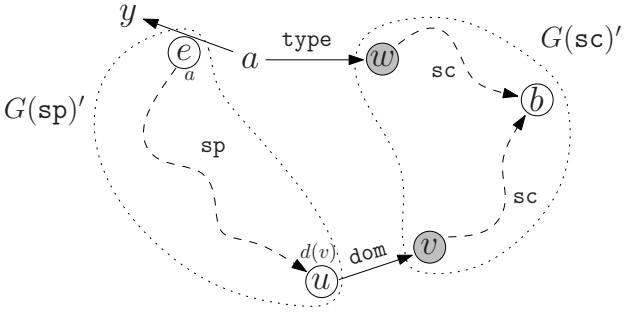

Fig. 2. Point 6 of the *Ground Entailment Algorithm*

Theorem 7. *Let (a, b, c) be a ground triple. The algorithm above can be used to test the entailment $G \models_{\rho\text{df}} (a, b, c)$ in time $\mathcal{O}(|G| \log |G|)$.*

Correctness and completeness of the algorithm follows from an inspection of the rules. The algorithm uses the rules in a bottom-up fashion. There are some subtleties in points 5 and 6. Point 5 follows from Lemma 2 and rule 2(a). The construction of $G(\texttt{sp})^*$ can be done in $|G| \log |G|$ steps: order $G_\emptyset$ and then while traversing $G(\texttt{sp})$ do binary search on $G_\emptyset$. For point 6 (see Figure 2) the crucial observation is that in $G(\texttt{sp})'$, if there is a path from a vertex marked a to a vertex u marked $d(v)$, then $G \models (a, u, y)$ for some y, and hence $G \models (a, \texttt{type}, v)$ using rule 4(a). Note that this checking takes time at most linear in $|G|$. From here, it is easy to see that the checking in $G(\texttt{sc})'$ will do the job.

Corollary 3. *Let H be a ground graph. Deciding if $G \models_{\rho\text{df}} H$ can be done in time $\mathcal{O}(|H| \cdot |G| \log |G|)$.*

The following result shows that the above algorithm cannot be essentially improved, in the sense that, any other algorithm for testing the ground entailment $G \models_{\rho\text{df}} H$ will take time proportional to $|H| \cdot |G| \log |G|$ in the worst case.

Theorem 8. *The problem of testing $G \models_{\rho\text{df}} t$ takes time $\Omega(|G| \log |G|)$.*

The bound is obtained by coding the problem of determining whether two sets are disjoint, which is a well known problem that needs $\Omega(n \log n)$ comparisons in the worst case [1]. The codification is as follows: Given the sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$, construct the RDF graph $G = \{(a_{i-1}, \texttt{sp}, a_i)\}_{2 \leq i \leq n} \cup \{(x, b_j, y)\}_{1 \leq j \leq n}$. Then, we have that $G \models (x, a_n, y)$ iff $A \cap B \neq \emptyset$.

6 Conclusions

We presented a streamlined fragment of RDFS which includes all the vocabulary that is relevant for describing data, avoiding vocabulary and semantics that theoretically corresponds to the definition of the structure of the language. We concentrated in studying the semantics, entailment, minimal proof systems, and algorithmic properties of this relevant fragment of RDFS. Our results show a viable proposal to lower the complexity of RDF data processing by using fragments of RDFS.

In this paper we have concentrated primarily on the ground dimension of RDF. Future work includes the refinement of our current results about the interplay between blank nodes semantics and the ground part. We are also working in the applications of our results to practical cases, as well as developing best practices for logical design of RDF specification based on the previous considerations.

Acknowledgments. Pérez was supported by Dirección de Investigación – Universidad de Talca, Gutierrez by Proyecto Enlace DI 2006, ENL 06/13, Universidad de Chile, and the three authors by Millennium Nucleus Center for Web Research, P04-067-F, Mideplan, Chile.

References

1. M. Ben-Or. *Lower bounds for algebraic computation trees. Proc. 15th Annual Symposium on Theory of Computing*, pp 80-86, 1983.
2. T. Berners-Lee. *Principles of Design. Personal Notes*, `http://www.w3.org/DesignIssues/Principles.html`.
3. Dan Brickley, Libby Miller. *FOAF Vocabulary Specification.* July 2005. `http://xmlns.com/foaf/0.1/`
4. J. de Bruijn, E. Franconi, S. Tessaris. *Logical Reconstruction of normative RDF.* In *OWLED 2005*, Galway, Ireland, November 2005
5. Victor Dalmau, P. G. Kolaitis, M. Vardi. *Constraint Satisfaction, Bounded Treewidth, and Finite–Variable Logics Proc. 8th Int. Conf. on Principles and Practice of Constraint Programming*, September, 2002.
6. Jeremy J. Carroll, Christian Bizer, Pat Hayes, Patrick Stickler, *Named graphs*, Journal of Web Semantics vol. 3, 2005, pp. 247 - 267
7. C. Gutierrez, C. Hurtado, A. O. Mendelzon, *Foundations of Semantic Web Databases*, Proceedings ACM Symposium on Principles of Database Systems (PODS), Paris, France, June 2004, pp. 95 - 106.
8. H. ter Horst. *C*ompleteness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Journal of Web Semantics, vol. 3, 2005.
9. Jean–Francois Baget, *RDF Entailment as a Graph Homomorphism*, In *ISWC 2005.*
10. Draltan Marin, *A Formalization of RDF (Applications de la Logique á la sémantique du web)*, École Polytechnique – Universidad de Chile, 2004. Technical Report Dept. Computer Science, Universidad de Chile, TR/DCC-2006-8. `http://www.dcc.uchile.cl/cgutierr/ftp/draltan.pdf`
11. E. Prud'hommeaux, A. Seaborne. *SPARQL Query Language for RDF.* W3C Working Draft, October 2006. `http://www.w3.org/TR/rdf-sparql-query/`.

12. *RDF/OWL Representation of WordNet.* Edit. Mark van Assem, Aldo Gangemi, Guus Schreiber. Working Draft, April 2006. `http://www.w3.org/2001/sw/BestPractices/WNET/wn-conversion`.
13. *Resource Description Framework (RDF) Model and Syntax Specification*, Edit. O. Lassila, R. Swick, Working draft, W3C, 1998.
14. *RDF/XML Syntax Specification (Revised) W3C Recommendation 10 February 2004*, Edit. D. Beckett
15. *RDF Semantics, W3C Recommendation 10 February 2004* Edit. P. Hayes
16. *RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004*, Edit. D. Brickley, R.V. Guha.
17. *RDF Concepts and Abstract Syntax, W3C Recommendation 10 February 2004*, Edit. G. Klyne, J. J. Carroll.
18. *RDF Primer, W3C Recommendation 10 February 2004*, Edit. F. Manola, E. Miller,
19. *Gene Ontology.* `http://www.geneontology.org/`

A Appendix: RDFS Semantics

To easy the job of the reader, we reproduce here the definitions and axioms of the normative semantics of RDF [15] consisting of a model theory and axiomatic triples. The set rdfsV stands for the RDFS vocabulary.

Definition 6 (cf. [15,10]). *The interpretation $\mathcal{I}$ is an* RDFS model *for an RDF graph G, denoted by $\mathcal{I} \models G$, iff $\mathcal{I}$ is an iterpretation over vocabulary* rdfsV $\cup$ universe(G) *that satisfies the RDF/S axiomatic triples [15,10] and the following semantic conditions:*

1. *Simple:*
   (a) *there exists a function $A : \mathbf{B} \to Res$ such that for each $(s, p, o) \in G$, $Int(p) \in Prop$ and $(Int_A(s), Int_A(o)) \in Ext(Int(p))$, where Int_A is the extension of Int using A.*
2. *RDF:*
   (a) $x \in Prop \Leftrightarrow (x, Int(\texttt{prop})) \in Ext(Int(\texttt{type}))$
   (b) *If $l \in$ universe(G) is a typed XML literal with lexical form w, then $Int(l)$ is the XML literal value of w, $Int(l) \in Lit$, and $(Int(l), Int(\texttt{xmlLit})) \in Ext(Int(\texttt{type}))$.*
3. *RDFS Classes:*
   (a) $x \in Res \Leftrightarrow x \in CExt(Int(\texttt{res}))$
   (b) $x \in Class \Leftrightarrow x \in CExt(Int(\texttt{class}))$
   (c) $x \in Lit \Leftrightarrow x \in CExt(Int(\texttt{literal}))$
4. *RDFS Subproperty:*
   (a) $Ext(Int(\texttt{sp}))$ *is transitive and reflexive over* $Prop$
   (b) *if* $(x, y) \in Ext(Int(\texttt{sp}))$ *then* $x, y \in Prop$ *and* $Ext(x) \subseteq Ext(y)$
5. *RDFS Subclass:*
   (a) $Ext(Int(\texttt{sc}))$ *is transitive and reflexive over* $Class$
   (b) *if* $(x, y) \in Ext(Int(\texttt{sc}))$ *then* $x, y \in Class$ *and* $CExt(x) \subseteq CExt(y)$
6. *RDFS Typing:*
   (a) $x \in CExt(y) \Leftrightarrow (x, y) \in Ext(Int(\texttt{type}))$
   (b) *if* $(x, y) \in Ext(Int(\texttt{dom}))$ *and* $(u, v) \in Ext(x)$ *then* $u \in CExt(y)$
   (c) *if* $(x, y) \in Ext(Int(\texttt{range}))$ *and* $(u, v) \in Ext(x)$ *then* $v \in CExt(y)$

7. *RDFS Additionals:*
   (a) *if* $x \in Class$ *then* $(x, Int(\texttt{res})) \in Ext(Int(\texttt{sc}))$.
   (b) *if* $x \in CExt(Int(\texttt{datatype}))$ *then* $(x, Int(\texttt{literal})) \in Ext(Int(\texttt{sc}))$
   (c) *if* $x \in CExt(Int(\texttt{contMP}))$ *then* $(x, Int(\texttt{member})) \in Ext(Int(\texttt{sp}))$

Now, given two graphs G *and* H *we say that* G RDFS entails H *and write* $G \models H$*, iff every RDFS model of* G *is also an RDFS model of* H*.*

Table 1. RDF/S vocabulary [15,10] with shortcuts in brackets. The first column shows built-in classes, second and third show built-in properties.

rdfs:Resource [res]	rdf:type [type]	rdfs:isDefinedBy [isDefined]
rdf:Property [prop]	rdfs:domain [dom]	rdfs:comment [comment]
rdfs:Class [class]	rdfs:range [range]	rdfs:label [label]
rdfs:Literal [literal]	rdfs:subClassOf [sc]	rdf:value [value]
rdfs:Datatype [datatype]	rdfs:subPropertyOf [sp]	rdf:nil [nil]
rdf:XMLLiteral [xmlLit]	rdf:subject [subj]	rdf:_1 [_1]
rdfs:Container [cont]	rdf:predicate [pred]	rdf:_2 [_2]
rdf:Statement [stat]	rdf:object [obj]	...
rdf:List [list]	rdfs:member [member]	rdf:_i [_i]
rdf:Alt [alt]	rdf:first [first]	...
rdf:Bag [bag]	rdf:rest [rest]	
rdf:Seq [seq]	rdfs:seeAlso [seeAlso]	
rdfs:ContainerMembershipProperty [contMP]		