Minimal Deductive Systems for RDF

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

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

Abstract This paper shows how one can do without several predicates and keywords of the RDFS theory, obtaining a simple language which preserves the original semantics. This approach is beneficial in at least two directions: (a) To have a simple abstract fragment of RDF easy to formalize and to reason about, which captures the essence of RDF; (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 RDF; 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 [15]. The motivations behind the development of RDF by the W3C were, among the most relevant for our discussion: providing information about Web resources and the systems that use them; to allow data to be processed outside the particular environment in which it was created, in a fashion that can work at Internet scale; combining data from several applications to arrive at new information; automated processing of Web information by software agents; to provide a world-wide lingua franca for these processes. 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 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 [10], Foaf [1] and Gene Ontology [17] 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 [5,6,2], 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 been studied in detail. Its description is given in [14] and its

semantics is defined in [13]. 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 [8,6]. 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* [10] and *Friend of a Friend* (FOAF) [1]. They both use only a proper subset of the RDFS vocabulary. 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`). The FOAF schema has no blank nodes. They are rarely used in the instances. 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 [9], which currently does not support RDFS vocabulary. 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.

2

Finally, regarding the point of complexity of entailment, not that due to the prospective size of RDF datasets, the exact bounds of entailment are crucial. There are two main building blocks of RDF when considering complexity: 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 [2,7,3]. These cases consider special forms of interaction between blank nodes that are very common in practice. In any case, there is a notion of normalized proof for RDFS entailment which makes it possible to treat the issue of blank nodes entailment in an 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. Proofs of all important results can be found in the Appendix.

2 RDF Semantics

Assume there are pairwise disjoint infinite sets **U** (RDF URI references), **B** (Blank nodes), and **L** (Literals). Through the paper we assume **U**, **B**, and **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 [4,9]– 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* **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 $\mu : \mathbf{UBL} \rightarrow \mathbf{UBL}$ 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 [13], and the mathematical formalization in [8] 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 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 [13,8] 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 [13,8] 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.

4

2.2   RDFS Vocabulary

The RDF specification includes a set of reserved words, the RDFS vocabulary (RDF Schema [14]) 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 [13], and (in brakets) 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" (see Table 2 Appendix A), 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 individuals 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 (see rule (3b)) 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:

$$\rho\mathrm{df} = \{\mathtt{sp}, \mathtt{sc}, \mathtt{type}, \mathtt{dom}, \mathtt{range}\}.$$

Definition 3. *Let G be a graph over ρdf. An interpretation $\mathcal{I}$ is a model of G under ρdf, denoted $\mathcal{I} \models_{\rho\mathrm{df}} G$, iff $\mathcal{I}$ is an interpretation over $\rho\mathrm{df} \cup \mathrm{universe}(G)$ 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) *$Ext(Int(\mathtt{sp}))$ is transitive and reflexive over $Prop$*
   (b) *if $(x, y) \in Ext(Int(\mathtt{sp}))$ then $x, y \in Prop$ and $Ext(x) \subseteq Ext(y)$*
3. *Subclass:*
   (a) *$Ext(Int(\mathtt{sc}))$ is transitive and reflexive over $Class$*
   (b) *if $(x, y) \in Ext(Int(\mathtt{sc}))$ then $x, y \in Class$ and $CExt(x) \subseteq CExt(y)$*
4. *Typing I:*
   (a) *$x \in CExt(y) \Leftrightarrow (x, y) \in Ext(Int(\mathtt{type}))$*
   (b) *if $(x, y) \in Ext(Int(\mathtt{dom}))$ and $(u, v) \in Ext(x)$ then $u \in CExt(y)$*
   (c) *if $(x, y) \in Ext(Int(\mathtt{range}))$ and $(u, v) \in Ext(x)$ then $v \in CExt(y)$*
5. *Typing II:*
   (a) *For each $\mathtt{e} \in \rho\mathrm{df}$, $Int(\mathtt{e}) \in Prop$.*
   (b) *if $(x, y) \in Ext(Int(\mathtt{dom}))$ then $x \in Prop$ and $y \in Class$.*
   (c) *if $(x, y) \in Ext(Int(\mathtt{range}))$ then $x \in Prop$ and $y \in Class$.*
   (d) *if $(x, y) \in Ext(Int(\mathtt{type}))$ then $y \in Class$.*

We define G entails H under ρdf, denoted $G \models_{\rho\mathrm{df}} H$, 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 [13,8], 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 $(\mathtt{dom}, \mathtt{dom}, \mathtt{prop})$, $(\mathtt{dom}, \mathtt{range}, \mathtt{class})$, $(\mathtt{range}, \mathtt{dom}, \mathtt{prop})$, $(\mathtt{range}, \mathtt{range}, \mathtt{class})$, and $(\mathtt{type}, \mathtt{range}, \mathtt{class})$, 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 [13,8], and let G and H be RDF graphs that do not mention RDFS vocabulary outside ρdf. Then*

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

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 $\mathtt{sp}$ 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 $\models_{\rho\mathrm{df}} t$. 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 $\mathtt{sp}$ and $\mathtt{sc}$. We will show that reflexivity for $\mathtt{sp}$ and $\mathtt{sc}$ 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 $\mathcal{I} \models^{\mathrm{nrx}}_{\rho\mathrm{df}} G$, 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,*

$$G \models_{\rho\mathrm{df}} H \quad \textit{iff} \quad G \models^{\mathrm{nrx}}_{\rho\mathrm{df}} H.$$

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 $\models^{\mathrm{nrx}}_{\rho\mathrm{df}}$ is that it does not entail axiomatic triples:

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

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, A, B, C, X, and Y are meta-variables representing elements in $\mathbf{UBL}$.

1. *Simple:*

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

---

2. *Subproperty:*

   (a) $\frac{(A,\mathsf{sp},B)\ \ (B,\mathsf{sp},C)}{(A,\mathsf{sp},C)}$ (b) $\frac{(A,\mathsf{sp},B)\ \ (X,A,Y)}{(X,B,Y)}$

3. *Subclass:*

   (a) $\frac{(A,\mathsf{sc},B)\ \ (B,\mathsf{sc},C)}{(A,\mathsf{sc},C)}$ (b) $\frac{(A,\mathsf{sc},B)\ \ (X,\mathsf{type},A)}{(X,\mathsf{type},B)}$

4. *Typing:*

   (a) $\frac{(A,\mathsf{dom},B)\ \ (X,a,Y)}{(X,\mathsf{type},B)}$ (b) $\frac{(A,\mathsf{range},B)\ \ (X,a,Y)}{(Y,\mathsf{type},B)}$

5. *Implicit Typing:*

   (a) $\frac{(A,\mathsf{dom},B)\ \ (C,\mathsf{sp},A)\ \ (X,C,Y)}{(X,\mathsf{type},B)}$ (b) $\frac{(A,\mathsf{range},B)\ \ (C,\mathsf{sp},A)\ \ (X,C,Y)}{(Y,\mathsf{type},B)}$

---

6. *Subproperty Reflexivity:*

   (a) $\frac{(X,A,Y)}{(A,\mathsf{sp},A)}$ (c) $\frac{}{(p,\mathsf{sp},p)}$ *for* $p \in \rho\mathrm{df}$

   (b) $\frac{(A,\mathsf{sp},B)}{(A,\mathsf{sp},A)\ \ (B,\mathsf{sp},B)}$ (d) $\frac{(A,p,X)}{(A,\mathsf{sp},A)}$ *for* $p \in \{\mathsf{dom}, \mathsf{range}\}$

7. *Subclass Reflexivity:*

   (a) $\frac{(A,\mathsf{sc},B)}{(A,\mathsf{sc},A)\ \ (B,\mathsf{sc},B)}$ (b) $\frac{(X,p,A)}{(A,\mathsf{sc},A)}$ *for* $p \in \{\mathsf{dom}, \mathsf{range}, \mathsf{type}\}$

Note 1 (On rules (5a) and (5b)). As noted in [8,6], the set of rules presented in [13] 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, sp, X), (X, dom, b), or (X, range, c). Here we solve the problem following the solution proposed by Marin [8] 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 $\vdash_{\rho\mathrm{df}}$ is sound and complete for $\models_{\rho\mathrm{df}}$, that is,*

$$G \vdash_{\rho\mathrm{df}} H \quad \textit{iff} \quad G \models_{\rho\mathrm{df}} H.$$

Corollary 2. *Define the proof system $\vdash_{\rho\mathrm{df}}^{\mathrm{nrx}}$ as $\vdash_{\rho\mathrm{df}}$ by droping rules of reflexivity (rules (6) and (7)). Then*

$$G \vdash_{\rho\mathrm{df}}^{\mathrm{nrx}} H \quad \textit{iff} \quad G \models_{\rho\mathrm{df}}^{\mathrm{nrx}} H.$$

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 $G \vdash_{\rho\mathrm{df}} H$. 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 $G \vdash_{\rho\mathrm{df}} H$, 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 $\vdash_{\rho\mathrm{df}}$ 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. Finally, we observe that, to be applied initially to G, any rule different of rule (1) must be an X-rule. Then, by induction follows that a proof of H from G can be constructed using only X-rules.
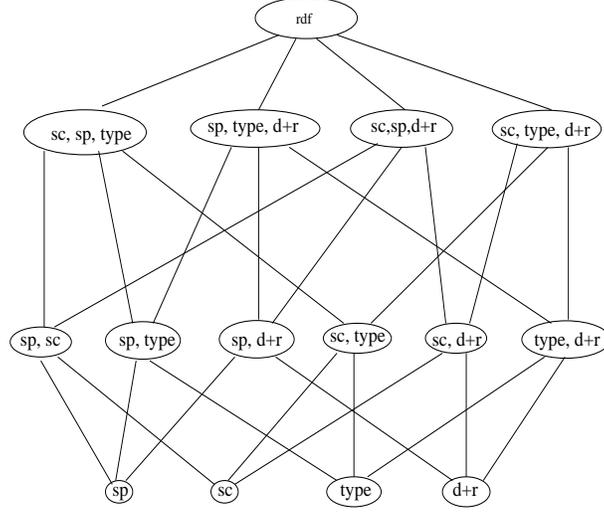
Figure 1. The lattice of fragments of ρdf.

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

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

Lemma 1. *Let a, b, c be ground terms with b not belonging to ρdf. Then: $G \models_{\rho\mathrm{df}}$ (a, b, c) iff $G|_{\{\mathrm{sp}, a, b, c\}} \models_{\rho\mathrm{df}} (a, b, c)$.*

Lemma 2. *Let $a, b \in \mathbf{UBL}$, 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 $G|_{\mathtt{range}} \models_{\rho\mathrm{df}} (a, \mathtt{range}, b)$.

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

Lemma 3 (sc,sp). *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 $G|_{\mathtt{sp}} \models_{\rho\mathrm{df}} (a, \mathtt{sp}, b)$.

10

Lemma 4. *let $G \models_{\rho\mathrm{df}} H$, then:*

1. *If $\mathtt{type} \notin \mathrm{voc}(H)$, then $G|_{\mathrm{voc}(H)} \models_{\rho\mathrm{df}} H$.*
2. *If $\mathtt{type} \in \mathrm{voc}(H)$, then $G|_{\mathrm{voc}(H) \cup \{\mathtt{dom}, \mathtt{range}, \mathtt{sp}\}} \models_{\rho\mathrm{df}} H$.*

5 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 the elements v which appear as subject or objects in triples of $G_{\mathtt{sp}}$, and in which (u, v) is an edge if and only if $(u, \mathtt{sp}, v) \in G_{\mathtt{sp}}$. Similar definition for $G(\mathtt{sc})$.

The naive approach to test the entailment $G \models H$ would be to consider the *closure* of G and check if H is included in it. The following results show that this procedure is too expensive in the worst case:

Theorem 6 (Size of Closure).

1. *The size of the closure of G is $\mathcal{O}(|G|^2)$.*
2. *The size of the closure of G is, in the worst case, no smaller than $\Omega(|G|^2)$.*

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. The lower bound follows from the example below.

Example 1 (lower bound for the closure). Consider the graph $\{(a_1, \mathtt{sp}, a_2), \ldots, (a_n, \mathtt{sp}, a_{n+1})\} \cup \{(x_1, a_1, y_n), \ldots, (x_n, an, y_n)\}$. The number of triples of the closure of this graph is $2n + 1 + \sum_{k=1}^{n} k$, that is order $\Omega(n^2)$.

The following algorithm presents a much better procedure to check 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 $\mathtt{sp}$) and rule 7 (for $\mathtt{sc}$).
5. IF $p \notin \rho$df 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 $(u, v, w) \in G_\emptyset$, if $v \in G(\mathtt{sp})$ then mark v with (u, w).
   IN Check in $G(\mathtt{sp})^*$ if there is a path from a vertex marked with (a, b)

which reaches p.

6. IF $p = \texttt{type}$ THEN

LET $G(\texttt{sp})'$ be the graph $G(\texttt{sp})$ with the following marks:
- For each triple $(u, \texttt{dom}, v) \in G_{\texttt{dom}}$, if $u \in G(\texttt{sp})$ mark the vertex u with $d(v)$.
- For each triple $(a, e, y) \in G_\emptyset$, if $e \in G(\texttt{sp})$, 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 $\texttt{range}$ instead of $\texttt{dom}$.
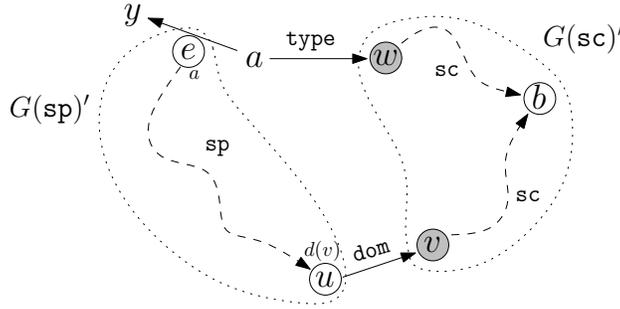
Figure 2. Point 6 of the *Ground Entailment Algorithm*

Theorem 7. *Let (a, b, c) be a ground triple. The algorithm above can be used to test the entailment $G \models_{\rho\mathrm{df}} (a, b, c)$ 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 1 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 $G \models_{\rho\mathrm{df}} H$ can be done in time $\mathcal{O}(|H| \cdot |G| \log |G|)$.*

The following result shows that the above algorithm cannot be essentially improved. The bound is obtained by coding the problem of determining if two sets are disjoint.

Proposition 2. *Testing $G \models t$ uses in the worst case $\Omega(|G| \log |G|)$ steps.*

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.

References

1. Dan Brickley, Libby Miller. *FOAF Vocabulary Specification*. July 2005. `http://xmlns.com/foaf/0.1/`
2. J. de Bruijn, E. Franconi, S. Tessaris. *Logical Reconstruction of normative RDF*. In *OWLED 2005*, Galway, Ireland, November 2005
3. 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.
4. Jeremy J. Carroll, Christian Bizer, Pat Hayes, Patrick Stickler, *Named graphs*, Journal of Web Semantics vol. 3, 2005, pp. 247 - 267
5. 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.
6. 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.
7. Jean–Francois Baget, *RDF Entailment as a Graph Homomorphism*, In *ISWC 2005*.
8. 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`
9. E. Prud'hommeaux, A. Seaborne. *SPARQL Query Language for RDF*. W3C Working Draft, October 2006. `http://www.w3.org/TR/rdf-sparql-query/`.

10. *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.
11. *Resource Description Framework (RDF) Model and Syntax Specification*, Edit. O. Lassila, R. Swick, Working draft, W3C, 1998.
12. *RDF/XML Syntax Specification (Revised) W3C Recommendation 10 February 2004*, Edit. D. Beckett
13. *RDF Semantics, W3C Recommendation 10 February 2004* Edit. P. Hayes
14. *RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004*, Edit. D. Brickley, R.V. Guha.
15. *RDF Concepts and Abstract Syntax, W3C Recommendation 10 February 2004*, Edit. G. Klyne, J. J. Carroll.
16. *RDF Primer, W3C Recommendation 10 February 2004*, Edit. F. Manola, E. Miller,
17. *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 [13] consisting of a model theory (Definition 6) and axiomatic triples (shown in Table 1). The set rdfsV stands for the RDFS vocabulary.

Definition 6 (cf. [13,8]). *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 [13,8] (see Table 2) 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) $x \in Prop \Leftrightarrow (x, Int(\mathtt{prop})) \in Ext(Int(\mathtt{type}))$
   (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 $(Int(l), Int(\mathtt{xmlLit})) \in Ext(Int(\mathtt{type}))$.*

3. *RDFS Classes:*
   (a) $x \in Res \Leftrightarrow x \in CExt(Int(\mathtt{res}))$
   (b) $x \in Class \Leftrightarrow x \in CExt(Int(\mathtt{class}))$
   (c) $x \in Lit \Leftrightarrow x \in CExt(Int(\mathtt{literal}))$

4. *RDFS Subproperty:*
   (a) $Ext(Int(\mathtt{sp}))$ *is transitive and reflexive over* $Prop$
   (b) *if $(x, y) \in Ext(Int(\mathtt{sp}))$ then $x, y \in Prop$ and $Ext(x) \subseteq Ext(y)$*

5. *RDFS Subclass:*
   (a) $Ext(Int(\mathtt{sc}))$ *is transitive and reflexive over* $Class$
   (b) *if $(x, y) \in Ext(Int(\mathtt{sc}))$ then $x, y \in Class$ and $CExt(x) \subseteq CExt(y)$*

6. *RDFS Typing:*
   (a) $x \in CExt(y) \Leftrightarrow (x, y) \in Ext(Int(\mathtt{type}))$
   (b) *if $(x, y) \in Ext(Int(\mathtt{dom}))$ and $(u, v) \in Ext(x)$ then $u \in CExt(y)$*

14

(c) if $(x, y) \in Ext(Int(\mathbf{range}))$ and $(u, v) \in Ext(x)$ then $v \in CExt(y)$

7. RDFS Additionals:

    (a) if $x \in Class$ then $(x, Int(\mathbf{res})) \in Ext(Int(\mathbf{sc}))$.

    (b) if $x \in CExt(Int(\mathbf{datatype}))$ then $(x, Int(\mathbf{literal})) \in Ext(Int(\mathbf{sc}))$

    (c) if $x \in CExt(Int(\mathbf{contMP}))$ then $(x, Int(\mathbf{member})) \in Ext(Int(\mathbf{sp}))$

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.

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**]		

Table 1. RDF/S vocabulary [13,8] with shortcuts in brakets. The first column shows built-in classes, second and third show built-in properties

(1) **Type**			(2) **Domain**			(3) **Range**			(4) **Subclass**		
(type,	type,	prop)	(type,	dom,	res)	(type,	range,	class)	(alt,	sc,	cont)
(subj,	type,	prop)	(dom,	dom,	prop)	(dom,	range,	class)	(bag,	sc,	cont)
(pred,	type,	prop)	(range,	dom,	prop)	(range,	range,	class)	(seq,	sc,	cont)
(obj,	type,	prop)	(sp,	dom,	prop)	(sp,	range,	prop)	(contMP,	sc,	prop)
(first,	type,	prop)	(sc,	dom,	class)	(sc,	range,	class)	(xmlLit,	sc,	literal)
(rest,	type,	prop)	(subj,	dom,	stat)	(subj,	range,	res)	(datatype,	sc,	class)
(value,	type,	prop)	(pred,	dom,	stat)	(pred,	range,	res)			
(_1,	type,	prop)	(obj,	dom,	stat)	(obj,	range,	res)	(4) **Subproperty**		
(_1,	type,	contMP)	(member,	dom,	res)	(member,	range,	res)	(isDefined, sp, seeAlso)		
(_2,	type,	prop)	(first,	dom,	list)	(first,	range,	res)			
(_2,	type,	contMP)	(rest,	dom,	list)	(rest,	range,	list)			
...			(seeAlso,	dom,	res)	(seeAlso,	range,	res)			
(_i,	type,	prop)	(isDefined,	dom,	res)	(isDefined,	range,	res)			
(_i,	type,	contMP)	(comment,	dom,	res)	(comment,	range,	literal)			
...			(label,	dom,	res)	(label,	range,	literal)			
(nil,	type,	prop)	(value,	dom,	res)	(value,	range,	res)			
(xmlLit,	type,	datatype)	(_1,	dom,	res)	(_1,	range,	res)			
			(_2,	dom,	res)	(_2,	range,	res)			
			...			...					
			(_i,	dom,	res)	(_i,	range,	res)			
			...			...					

Table 2. RDF/S axiomatic triples [13,8]

B Proofs of Section 3

B.1 Proof of Theorem 1

In the proof of this Theorem we use Definition 6 of Appendix A for RDFS models. We make the proof assuming that RDFS models do not impose conditions about XML typed literals (this is not a serious restriction an the reader will note that the proof can be easily extended).

Proof. $\Leftarrow$) Let $\mathcal{I}$ be an RDFS model of G, that is, $\mathcal{I}$ satisfies all the conditions in Definition 6 for G. Then, because $\mathcal{I}$ satisfies conditions 1, 4, 5, and 6 of Definition 6, $\mathcal{I}$ interpret every element in ρdf as property names, and also satisfies the axiomatic triples $(\mathsf{dom}, \mathsf{dom}, \mathsf{prop})$, $(\mathsf{dom}, \mathsf{range}, \mathsf{class})$, $(\mathsf{range}, \mathsf{dom}, \mathsf{prop})$, $(\mathsf{range}, \mathsf{range}, \mathsf{class})$, and $(\mathsf{type}, \mathsf{range}, \mathsf{class})$, $\mathcal{I}$ satisfies all conditions in Defintion 3, and then $\mathcal{I}$ is also an ρdf model for G. Now $\mathcal{I}$ is an ρdf model for H that satisfies all the conditions of Definition 6 and then $\mathcal{I}$ is also an RDFS model for H, completing this part of the proof.

$\Rightarrow$) Let $\mathcal{I} = (Res, Prop, Class, Ext, CExt, Lit, Int)$ be a model of G under ρdf. We will construct an RDFS model $\mathcal{I}'$ of G from $\mathcal{I}$. Suppose first that for every $\mathsf{e} \in \text{rdfsV}$ there is an element x_e that will be used to interpret e in $\mathcal{I}'$, and such that in $\mathcal{I}$, $Int(\mathsf{e}) = x_\mathsf{e}$ for every $\mathsf{e} \in \rho$df. Note also that, because $\mathcal{I}$ is an interpretation under ρdf, then for every $\mathsf{e} \in \text{rdfsV} - \rho$df, $Int(\mathsf{e})$ is not defined. Let Ax be the set of all RDF/S axiomatic triples [13,8] (see Table 2).

Consider the interpretation $\mathcal{I}' = (Res', Prop', Class', Ext', CExt', Lit', Int')$ constructed in the following way:

- $Res' = Res \cup Prop \cup \{x_\mathsf{e} \mid \mathsf{e} \in \text{rdfsV}\} \cup \{l\}$.
- $Prop' = Prop \cup \{x_\mathsf{e} \mid \mathsf{e} \in \rho\text{df}\} \cup$
  $\{x_\mathsf{e} \mid (\mathsf{e}, \mathsf{type}, \mathsf{prop}) \in Ax\} \cup$
  $\{x_\mathsf{e} \mid (\mathsf{e}, \mathsf{sp}, y), (z, \mathsf{sp}, \mathsf{e}), (\mathsf{e}, \mathsf{dom}, u), \text{ or } (\mathsf{e}, \mathsf{range}, v) \in Ax\} \cup$
  $\{x \mid (x, y) \in Ext(x_\mathsf{sp}), (z, x) \in Ext(x_\mathsf{sp}),$
  $\quad (x, u) \in Ext(x_\mathsf{dom}), \text{ or } (x, v) \in Ext(x_\mathsf{range})\}$.
- $Class' = Class \cup$
  $\{x_\mathsf{e} \mid (y, \mathsf{type}, \mathsf{e}) \in Ax\} \cup$
  $\{x_\mathsf{e} \mid (\mathsf{e}, \mathsf{sc}, y), (z, \mathsf{sc}, \mathsf{e}), (u, \mathsf{dom}, \mathsf{e}), \text{ or } (v, \mathsf{range}, \mathsf{e}) \in Ax\} \cup$
  $\{x \mid (y, x) \in Ext(x_\mathsf{type})\} \cup$
  $\{x \mid (x, y) \in Ext(x_\mathsf{sc}), (z, x) \in Ext(x_\mathsf{sc}),$
  $\quad (u, x) \in Ext(x_\mathsf{dom}), \text{ or } (v, x) \in Ext(x_\mathsf{range})\}$.
- $Lit' = Lit$.
- Int' is such that for every $\mathsf{e} \in \text{rdfsV}$, $Int'(\mathsf{e}) = x_\mathsf{e}$, and $Int'(x) = Int(x)$ in other case.
- Ext' is an extension function such that:
  - $Ext'(x_\mathsf{type}) =$
    $Ext(x_\mathsf{type}) \cup$
    $\{(x_\mathsf{s}, x_\mathsf{o}) \mid (\mathsf{s}, \mathsf{type}, \mathsf{o}) \in Ax\} \cup$
    $\{(y, x_\mathsf{res}) \mid y \in Res'\} \cup$

16

$$\{(y, x_{\mathtt{prop}}) \mid y \in Prop'\} \cup$$
$$\{(y, x_{\mathtt{class}}) \mid y \in Class'\} \cup$$
$$\{(y, x_{\mathtt{literal}}) \mid y \in Lit'\} \cup$$
$$\{(x, y) \mid x \in Res', (x_{\mathtt{e}}, y) \in Ext(x_{\mathtt{dom}}) \cup Ext(x_{\mathtt{range}}) \text{ with } \mathtt{e} \in \rho\mathrm{df}\}.$$

- $Ext'(x_{\mathtt{dom}}) =$
  $Ext(x_{\mathtt{dom}}) \cup$
  $\{(x_{\mathtt{s}}, x_{\mathtt{o}}) \mid (\mathtt{s}, \mathtt{dom}, \mathtt{o}) \in Ax\}.$

- $Ext'(x_{\mathtt{range}}) =$
  $Ext(x_{\mathtt{range}}) \cup$
  $\{(x_{\mathtt{s}}, x_{\mathtt{o}}) \mid (\mathtt{s}, \mathtt{range}, \mathtt{o}) \in Ax\}.$

- $Ext'(x_{\mathtt{sc}}) =$
  $Ext(x_{\mathtt{sc}}) \cup$
  $\{(x_{\mathtt{s}}, x_{\mathtt{o}}) \mid (\mathtt{s}, \mathtt{sc}, \mathtt{o}) \in Ax\} \cup$
  $\{(x, x) \mid x \in Class'\} \cup$
  $\{(y, x_{\mathtt{res}}) \mid y \in Class'\}.$

- $Ext'(x_{\mathtt{sp}}) =$
  $Ext(x_{\mathtt{sp}}) \cup$
  $\{(x_{\mathtt{s}}, x_{\mathtt{o}}) \mid (\mathtt{s}, \mathtt{sp}, \mathtt{o}) \in Ax\} \cup$
  $\{(x, x) \mid x \in Prop'\} \cup$
  $\{(x_{\_1}, x_{\mathtt{member}}), (x_{\_2}, x_{\mathtt{member}}), \ldots\}.$

- $Ext'(x_{\mathtt{e}}) = \emptyset$ for every $x_{\mathtt{e}} \in Prop'$ such that $\mathtt{e} \in \mathrm{rdfsV} - \rho\mathrm{df}$.

- $Ext'(x) = Ext(x)$ in all other cases.

– $CExt'$ is such that:

- $CExt'(x_{\mathtt{res}}) = Res'.$
- $CExt'(x_{\mathtt{prop}}) = Prop'.$
- $CExt'(x_{\mathtt{class}}) = Class'.$
- $CExt'(x_{\mathtt{literal}}) = Lit'.$
- $CExt'(x_{\mathtt{contMP}}) = \{x_{\_1}, x_{\_2}, \ldots\}.$
- $CExt'(x_{\mathtt{datatype}}) = \{x_{\mathtt{xmlLit}}\}.$
- $CExt'(x_{\mathtt{e}}) = \emptyset$ for $\mathtt{e} \in \{\mathtt{xmlLit}, \mathtt{cont}, \mathtt{alt}, \mathtt{bag}, \mathtt{seq}, \mathtt{list}, \mathtt{stat}\}.$
- $CExt'(x) = CExt(x) \cup Res'$ if $(x_{\mathtt{e}}, x) \in Ext(x_{\mathtt{dom}}) \cup Ext(x_{\mathtt{range}})$ for $\mathtt{e} \in \rho\mathrm{df}$.
- $CExt'(x) = CExt(x)$ in all other cases.

Note that $\mathcal{I}'$ is well defined in the sense that every one of its components is defined in terms of notions defined before.

Now we prove that $\mathcal{I}'$ is an RDFS model of G. First note that for every RDF/S axiomatic triple $(\mathtt{s}, \mathtt{p}, \mathtt{o})$ we have that $\mathtt{p} \in \rho\mathrm{df}$. Now by the construction of $Prop'$, Int', and Ext', for every RDF/S axiomatic triple $(\mathtt{s}, \mathtt{p}, \mathtt{o})$, we have that $Int'(\mathtt{p}) = x_{\mathtt{p}} \in Prop'$ and $(Int'(\mathtt{s}), Int'(\mathtt{o})) = (x_{\mathtt{s}}, x_{\mathtt{o}}) \in Ext'(x_{\mathtt{p}}) = Ext'(Int'(\mathtt{p}))$, and then $\mathcal{I}'$ satisfies all RDF/S axiomatic triples.

Now we prove that $\mathcal{I}'$ satisfies also all the conditions en Definition 6. First observe that $\mathcal{I}$ satisfies conditions 1, 4, 5, and 6 of Definition 6 for G, because $\mathcal{I}$ is an ρdf model for G. Now for $\mathcal{I}'$:

1. Simple:
   (a) For every $\mathsf{e} \in \rho\mathrm{df}$ we have that $Int'(\mathsf{e}) = x_{\mathsf{e}} = Int(\mathsf{e})$ and $Ext(x_{\mathsf{e}}) \subseteq Ext'(x_{\mathsf{e}})$, and Int' and Ext' are defined exactly as Int and Ext in all other cases. Note also that G does not mention RDFS vocabulary outside ρdf. Then, for every triple $(s, p, o) \in G$ we have that $(Int'(s), Int'(o)) = (Int(s), Int(o)) \in Ext(Int(p)) \subseteq Ext'(Int(p)) = Ext'(Int'(p))$. Then $\mathcal{I}'$ satisfies this condition for G.

2. RDF:
   (a) Note that G does not mention prop, then $\mathcal{I}$ does not interepret prop and then there is no y such that $(y, x_{\mathsf{prop}}) \in Int(x_{\mathsf{type}})$ in $\mathcal{I}$. Then by definition of $Ext'(x_{\mathsf{type}})$ in $\mathcal{I}'$ we have that $(y, x_{\mathsf{prop}}) \in Ext'(x_{\mathsf{type}})$ iff $y \in Prop'$, and then $\mathcal{I}'$ satifies this condition for G.

3. RDFS Classes:
   (a) By the construction of $\mathcal{I}'$ we have $CExt'(x_{\mathsf{res}}) = Res'$.
   (b) By the construction of $\mathcal{I}'$ we have $CExt'(x_{\mathsf{class}}) = Class'$.
   (c) By the construction of $\mathcal{I}'$ we have $CExt'(x_{\mathsf{literal}}) = Lit'$.

4. RDFS Subproperty:
   (a) By the construction of $\mathcal{I}'$ we have that $Ext'(x_{\mathsf{sp}})$ is reflexive over $Prop'$. Now, note that the only axiomatic triple that mention sp in its predicate position is $(\mathsf{isDefined}, \mathsf{sp}, \mathsf{seeAlso})$. Then we must only prove that $Ext(x_{\mathsf{sp}}) \cup \{(x_{\mathsf{isDefined}}, x_{\mathsf{seeAlso}}), (x_{\_1}, x_{\mathsf{member}}), (x_{\_2}, x_{\mathsf{member}}), \ldots\}$ is a transitive relation, which is a direct consequence of the fact that $Ext(x_{\mathsf{sp}})$ is transitive and G does not mention $\mathsf{isDefined}$, nor $\mathsf{seeAlso}$, nor $\_\mathsf{i}$ for any i.
   (b) Let $(x, y) \in Ext'(x_{\mathsf{sp}}) = Ext(x_{\mathsf{sp}}) \cup \{(x_{\mathsf{isDefined}}, x_{\mathsf{seeAlso}})\} \cup \{(x, x) \mid x \in Prop'\} \cup \{(x_{\_1}, x_{\mathsf{member}}), (x_{\_2}, x_{\mathsf{member}}), \ldots\}$. If $(x, y) \in Ext(x_{\mathsf{sp}})$ then the condition holds because $\mathcal{I}$ satisfies this condition. For $(x_{\mathsf{isDefined}}, x_{\mathsf{seeAlso}})$ we have that $x_{\mathsf{isDefined}}, x_{\mathsf{seeAlso}} \in Prop'$ and $Ext'(x_{\mathsf{isDefined}}) = \emptyset \subseteq Ext'(x_{\mathsf{seeAlso}})$. For $(x_{\_\mathsf{i}}, x_{\mathsf{member}})$ we have that $x_{\_\mathsf{i}}, x_{\mathsf{member}} \in Prop'$ and $Ext'(x_{\_\mathsf{i}}) = \emptyset \subseteq Ext'(x_{\mathsf{member}})$. Finally, for $(x, x) \in Ext'(x_{\mathsf{sp}})$ we have $x \in Prop'$ and then the condition holds.

5. RDFS Subclass:
   (a) By the construction of $\mathcal{I}'$ we have that $Ext'(x_{\mathsf{sc}})$ is reflexive over $Class'$. Now, note that for every pair of axiomatic triples $(\mathsf{c_1}, \mathsf{sc}, \mathsf{c_2})$, $(\mathsf{c_3}, \mathsf{sc}, \mathsf{c_4})$, we have that $\mathsf{c_2} \neq \mathsf{c_3}$, and $\mathsf{c_1}, \mathsf{c_2}, \mathsf{c_3}, \mathsf{c_4} \in \mathrm{rdfsV} - \rho\mathrm{df}$ (see Table 2). Consider $(x, y), (y, z) \in Ext'(x_{\mathsf{sc}})$: if $x = y$ or $y = z$ then $(x, z) \in Ext'(x_{\mathsf{sc}})$; if $x \neq y$ and $y \neq z$, by the previous observation and the fact that G does not mention RDFS vocabulary outside ρdf, we have that $(x, y), (y, z) \in Ext(x_{\mathsf{sc}})$ and then by transitivity of $Ext(x_{\mathsf{sc}})$ we have that $(x, z) \in Ext(x_{\mathsf{sc}}) \subseteq Ext'(x_{\mathsf{sc}})$. Finally $Ext'(x_{\mathsf{sc}})$ is a transitive relation.

18

(b) Let $(x, y) \in Ext'(x_{\mathsf{sc}}) = Ext(x_{\mathsf{sc}}) \cup \{(x_{\mathsf{s}}, x_{\mathsf{o}}) \mid (\mathsf{s}, \mathsf{sc}, \mathsf{o}) \in Ax\} \cup \{(x, x) \mid x \in Class'\} \cup \{(y, x_{\mathsf{res}}) \mid \text{ for every } y \in Class'\}$. If $(x, y) \in Ext(x_{\mathsf{sc}})$ then the condition holds because $\mathcal{I}$ satisfies this condition. Now, note that for every axiomatic triple $(\mathsf{c}_1, \mathsf{sc}, \mathsf{c}_2)$, by the construction of $\mathcal{I}'$ we have that $x_{\mathsf{c}_1}, x_{\mathsf{c}_2} \in Class'$ and $CExt'(x_{\mathsf{c}_1}) = \emptyset$ except for the case when c_1 is contMP or $\mathsf{datatype}$. Then, if $(x, y) \in \{(x_{\mathsf{s}}, x_{\mathsf{o}}) \mid (\mathsf{s}, \mathsf{sc}, \mathsf{o}) \in Ax\}$ with $x \neq x_{\mathsf{contMP}}$ and $x \neq x_{\mathsf{datatype}}$ we have that $x, y \in Class'$ and $CExt'(x) = \emptyset \subseteq CExt'(y)$. For the case in which $x = x_{\mathsf{contMP}}$ we have $y = x_{\mathsf{prop}}$, and by the construction of $\mathcal{I}'$ we have $CExt'(x) = \{x\_1, x\_2, \ldots\} \subseteq Prop' = CExt'(x_{\mathsf{prop}}) = CExt'(y)$. For the case in which $x = x_{\mathsf{datatype}}$ we have $y = x_{\mathsf{class}}$, and by the construction of $\mathcal{I}'$ we have $CExt'(x) = \{x_{\mathsf{xmlLit}}\} \subseteq Class' = CExt'(x_{\mathsf{class}}) = CExt'(y)$. If $(x, y) \in \{(y, x_{\mathsf{res}}) \mid y \in Class'\}$ we have that $x, y = x_{\mathsf{res}} \in Class'$, and by the construction of $\mathcal{I}'$, $CExt'(x) \subseteq Res' = CExt'(x_{\mathsf{res}}) = CExt'(y)$ (note that Res' is a superset of $Prop'$, $Class'$, and Lit', and that in $\mathcal{I}$ for every $x \in Class$ we have $CExt(x) \subseteq Res$). Finally, if $(x, y) \in \{(x, x) \mid x \in Class'\}$ then $x = y \in Class'$ and $CExt'(x) \subseteq CExt'(y)$, completing the proof that $\mathcal{I}'$ satisfies this condition.

6. RDFS Typing:

(a) ($\Rightarrow$) Let $x \in CExt'(y)$, we have several cases: First note that $y \neq x_{\mathsf{e}}$ for every $\mathsf{e} \in \{\mathsf{xmlLit}, \mathsf{cont}, \mathsf{alt}, \mathsf{bag}, \mathsf{seq}, \mathsf{list}, \mathsf{stat}\}$ because in these cases $CExt'(y) = \emptyset$. If $y = x_{\mathsf{e}}$ for $\mathsf{e} \in \{\mathsf{res}, \mathsf{prop}, \mathsf{class}, \mathsf{literal}\}$ then we have $(x, y) \in Ext'(x_{\mathsf{type}})$ by the construction of $Ext'(x_{\mathsf{type}})$ in $\mathcal{I}'$. If $y = x_{\mathsf{contMP}}$ then $x = x\_i$ for some $\_i$, and because for every $\_i$ there is an axiomatic triple $(\_i, \mathsf{type}, \mathsf{contMP})$, we have that $(x, y) \in Ext'(x_{\mathsf{type}})$. If $y = x_{\mathsf{datatype}}$ then $x = x_{\mathsf{xmlLit}}$, and because there is an axiomatic triple $(\mathsf{xmlLit}, \mathsf{type}, \mathsf{datatype})$, we have that $(x, y) \in Ext'(x_{\mathsf{type}})$. Now if y is such that $(x_{\mathsf{e}}, y) \in Ext(x_{\mathsf{dom}}) \cup Ext(x_{\mathsf{range}})$ for $\mathsf{e} \in \rho\mathrm{df}$, then $x \in CExt'(y) = Res'$ and then by the construction of $Ext'(x_{\mathsf{type}})$ we have $(x, y) \in Ext'(x_{\mathsf{type}})$. In other case $CExt'(y) = CExt(y)$ and then, because $\mathcal{I}$ satisfies this condition, we have that $(x, y) \in Ext(x_{\mathsf{type}}) \subseteq Ext'(x_{\mathsf{type}})$.

($\Leftarrow$) Now, if we consider $(x, y) \in Ext'(x_{\mathsf{type}})$, we have several cases. If $(x, y) \in \{(y, x_{\mathsf{res}}) \mid y \in Res'\} \cup \{(y, x_{\mathsf{class}}) \mid y \in Class'\} \cup \{(y, x_{\mathsf{prop}}) \mid y \in Prop'\} \cup \{(y, x_{\mathsf{literal}}) \mid y \in Lit'\}$ then by the construction of $\mathcal{I}'$ we have $x \in CExt'(y)$. If $(x, y) \in Ext(x_{\mathsf{type}})$ then, because $\mathcal{I}$ satisfies this condition and G does not mentions RDFS vocabulary outside $\rho\mathrm{df}$ we have $x \in CExt(y) \subseteq CExt'(y)$. If $(x, y) \in \{(x_{\mathsf{s}}, x_{\mathsf{o}}) \mid (\mathsf{s}, \mathsf{type}, \mathsf{o}) \in Ax\}$ then by the construction of $Prop'$, $CExt'(x_{\mathsf{prop}})$, $CExt'(x_{\mathsf{contMP}})$, and $CExt'(x_{\mathsf{xmlLit}})$, and the specific axiomatic triples that have type as predicate (see Table 2), we have $x \in CExt'(y)$. If (x, y) is such that $(x_{\mathsf{e}}, y) \in Ext(x_{\mathsf{dom}}) \cup Ext(x_{\mathsf{range}})$ with $\mathsf{e} \in \rho\mathrm{df}$, we have by construction of $CExt'$ that $CExt'(y) = Res'$ and then because $x \in Res'$ we have $x \in CExt'(y)$, completing the proof.

(b) Let $(x, y) \in Ext'(x_{\mathsf{dom}})$ and $(u, v) \in Ext'(x)$. First note that $x \neq x_{\mathsf{e}}$ for every $\mathsf{e} \in \mathrm{rdfsV} - \rho\mathrm{df}$ with $x_{\mathsf{e}} \in Prop'$, because in these cases $Ext'(x) =$

$\emptyset$. Also note that if $(x, y) \in Ext(x_{\mathsf{dom}})$ and $(u, v) \in Ext(x)$ then, because $\mathcal{I}$ satisfies this condition we have that $u \in CExt(y)$. Additionally note that Ext' is different to Ext only in elements x_{e} with $\mathsf{e} \in \rho\mathrm{df}$, so all remaining cases that left to be checked are the ones in which $(u, v) \in Ext'(x_{\mathsf{e}})$ with $\mathsf{e} \in \rho\mathrm{df}$. We consider now all the remaining cases.

- $x = x_{\mathsf{type}}$: If $(x_{\mathsf{type}}, y) \in Ext'(x_{\mathsf{dom}})$, and $(u, v) \in Ext'(x_{\mathsf{type}})$ we must prove that $u \in CExt'(y)$. First, if $(x_{\mathsf{type}}, y) \in Ext'(x_{\mathsf{dom}})$ then $(x_{\mathsf{type}}, y) \in Ext(x_{\mathsf{dom}})$ or $y = x_{\mathsf{res}}$ by the axiomatic triple $(\mathsf{type}, \mathsf{dom}, \mathsf{res})$. If $y = x_{\mathsf{res}}$ and $(u, v) \in Ext'(x_{\mathsf{type}})$, then the condition holds because $u \in Res' = CExt'(x_{\mathsf{res}}) = CExt'(y)$. Now suppose that $(x_{\mathsf{type}}, y) \in Ext(x_{\mathsf{dom}})$, then by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and then because $u \in Res'$ we obtain $u \in CExt'(y)$.
- $x = x_{\mathsf{dom}}$: If $(x_{\mathsf{dom}}, y) \in Ext'(x_{\mathsf{dom}})$, and $(u, v) \in Ext'(x_{\mathsf{dom}})$ we must prove that $u \in CExt'(y)$. First, if $(x_{\mathsf{dom}}, y) \in Ext'(x_{\mathsf{dom}})$ then $(x_{\mathsf{dom}}, y) \in Ext(x_{\mathsf{dom}})$ or $y = x_{\mathsf{prop}}$ by the axiomatic triple $(\mathsf{dom}, \mathsf{dom}, \mathsf{prop})$. If $y = x_{\mathsf{prop}}$ and $(u, v) \in Ext'(x_{\mathsf{dom}})$, we have two cases: if $(u, v) \in \{(x_{\mathsf{s}}, x_{\mathsf{o}}) \mid (\mathsf{s}, \mathsf{dom}, \mathsf{o}) \in Ax\}$ then by the construction of $\mathcal{I}'$ we have $u \in Prop' = CExt'(x_{\mathsf{prop}}) = CExt'(y)$; if $(u, v) \in Ext(x_{\mathsf{dom}})$ then by the construction of $Prop'$ we have $u \in Prop'$. Now if $(x_{\mathsf{dom}}, y) \in Ext(x_{\mathsf{dom}})$, then by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and then because $u \in Res'$ we obtain $u \in CExt'(y)$.
- $x = x_{\mathsf{range}}$: If $(x_{\mathsf{range}}, y) \in Ext'(x_{\mathsf{dom}})$, and $(u, v) \in Ext'(x_{\mathsf{range}})$ we must prove that $u \in CExt'(y)$. First, if $(x_{\mathsf{range}}, y) \in Ext'(x_{\mathsf{dom}})$ then $(x_{\mathsf{range}}, y) \in Ext(x_{\mathsf{dom}})$ or $y = x_{\mathsf{prop}}$ by the axiomatic triple $(\mathsf{range}, \mathsf{dom}, \mathsf{prop})$. If $y = x_{\mathsf{prop}}$ and $(u, v) \in Ext'(x_{\mathsf{range}})$, we have two cases: if $(u, v) \in \{(x_{\mathsf{s}}, x_{\mathsf{o}}) \mid (\mathsf{s}, \mathsf{dom}, \mathsf{o}) \in Ax\}$ then by the construction of $\mathcal{I}'$ we have $u \in Prop' = CExt'(x_{\mathsf{prop}}) = CExt'(y)$; if $(u, v) \in Ext(x_{\mathsf{range}})$ then by the construction of $Prop'$ we have $u \in Prop'$. Now if $(x_{\mathsf{range}}, y) \in Ext(x_{\mathsf{dom}})$, then by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and then because $u \in Res'$ we obtain $u \in CExt'(y)$.
- $x = x_{\mathsf{sp}}$: If $(x_{\mathsf{sp}}, y) \in Ext'(x_{\mathsf{dom}})$, and $(u, v) \in Ext'(x_{\mathsf{sp}})$ we must prove that $u \in CExt'(y)$. First, if $(x_{\mathsf{sp}}, y) \in Ext'(x_{\mathsf{dom}})$ then $(x_{\mathsf{sp}}, y) \in Ext(x_{\mathsf{dom}})$ or $y = x_{\mathsf{prop}}$ by the axiomatic triple $(\mathsf{sp}, \mathsf{dom}, \mathsf{prop})$. If $y = x_{\mathsf{prop}}$ and $(u, v) \in Ext'(x_{\mathsf{sp}})$, we have several cases: if $(u, v) \in \{(x_{\mathsf{s}}, x_{\mathsf{o}}) \mid (\mathsf{s}, \mathsf{sp}, \mathsf{o}) \in Ax\}$ then by the construction of $\mathcal{I}'$ we have $u \in Prop' = CExt'(x_{\mathsf{prop}}) = CExt'(y)$; if $(u, v) \in \{(x, x) \mid x \in Prop'\}$ then $u \in Prop' = CExt'(x_{\mathsf{prop}}) = CExt'(y)$; if $(u, v) = (x_{\_\mathsf{i}}, x_{\mathsf{member}})$ for some $\_\mathsf{i}$, then by the construction of $Prop'$ because there is an axiomatic triple $(\_\mathsf{i}, \mathsf{type}, \mathsf{prop})$ for every $\_\mathsf{i}$ we have $u \in Prop'$; and if $(u, v) \in Ext(x_{\mathsf{prop}})$ then by the construction of $Prop'$ we have $u \in Prop'$. Now if $(x_{\mathsf{prop}}, y) \in Ext(x_{\mathsf{dom}})$, then by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and then because $u \in Res'$ we obtain $u \in CExt'(y)$.

- $x = x_{\mathsf{sc}}$: If $(x_{\mathsf{sc}}, y) \in Ext'(x_{\mathsf{dom}})$, and $(u, v) \in Ext'(x_{\mathsf{sc}})$ we must prove that $u \in CExt'(y)$. First, if $(x_{\mathsf{sc}}, y) \in Ext'(x_{\mathsf{dom}})$ then $(x_{\mathsf{sc}}, y) \in Ext(x_{\mathsf{dom}})$ or $y = x_{\mathsf{class}}$ by the axiomatic triple $(\mathsf{sc}, \mathsf{dom}, \mathsf{class})$. If $y = x_{\mathsf{class}}$ and $(u, v) \in Ext'(x_{\mathsf{sc}})$, we have several cases: if $(u, v) \in \{(x_{\mathsf{s}}, x_{\mathsf{o}}) \mid (\mathsf{s}, \mathsf{sc}, \mathsf{o}) \in Ax\}$ then by the construction of $\mathcal{I}'$ we have $u \in Class' = CExt'(x_{\mathsf{class}}) = CExt'(y)$; if $(u, v) \in \{(x, x) \mid x \in Class'\}$ then $u \in Class' = CExt'(x_{\mathsf{class}}) = CExt'(y)$; if $(u, v) \in \{(x, x_{\mathsf{res}}) \mid x \in Class'\}$ then $u \in Class' = CExt'(x_{\mathsf{class}}) = CExt'(y)$; and if $(u, v) \in Ext(x_{\mathsf{class}})$ then by the construction of $Class'$ we have $u \in Class'$. Now if $(x_{\mathsf{class}}, y) \in Ext(x_{\mathsf{dom}})$, then by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and then because $u \in Res'$ we obtain $u \in CExt'(y)$.

Then, in all cases $\mathcal{I}'$ satisfies this condition for G.

(c) Let $(x, y) \in Ext'(x_{\mathsf{range}})$ and $(u, v) \in Ext'(x)$, we must prove that $v \in CExt'(y)$. The same observations for the previous case hold here, so we must concentrate in cases in which $(u, v) \in Ext'(x_{\mathsf{e}})$ with $\mathsf{e} \in \rho\mathrm{df}$.

- $x = x_{\mathsf{type}}$: the same proof for x_{type} in the previous condition works here considering v instead of u and changing x_{dom} with x_{range}.

- $x = x_{\mathsf{dom}}$: If $(x_{\mathsf{dom}}, y) \in Ext'(x_{\mathsf{range}})$, and $(u, v) \in Ext'(x_{\mathsf{dom}})$ we must prove that $v \in CExt'(y)$. First, if $(x_{\mathsf{dom}}, y) \in Ext'(x_{\mathsf{range}})$ then $(x_{\mathsf{dom}}, y) \in Ext(x_{\mathsf{range}})$ or $y = x_{\mathsf{class}}$ by the axiomatic triple $(\mathsf{dom}, \mathsf{range}, \mathsf{class})$. If $y = x_{\mathsf{class}}$ and $(u, v) \in Ext'(x_{\mathsf{dom}})$, we have two cases: if $(u, v) \in \{(x_{\mathsf{s}}, x_{\mathsf{o}}) \mid (\mathsf{s}, \mathsf{range}, \mathsf{o}) \in Ax\}$ then by the construction of $\mathcal{I}'$ we have $v \in Class' = CExt'(x_{\mathsf{class}}) = CExt'(y)$; if $(u, v) \in Ext(x_{\mathsf{range}})$ then by the construction of $Class'$ we have $v \in Class'$. Now if $(x_{\mathsf{dom}}, y) \in Ext(x_{\mathsf{range}})$, then by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and then because $v \in Res'$ we obtain $v \in CExt'(y)$.

- $x = x_{\mathsf{range}}$: If $(x_{\mathsf{range}}, y) \in Ext'(x_{\mathsf{range}})$, and $(u, v) \in Ext'(x_{\mathsf{range}})$ we must prove that $v \in CExt'(y)$. First, if $(x_{\mathsf{range}}, y) \in Ext'(x_{\mathsf{range}})$ then $(x_{\mathsf{range}}, y) \in Ext(x_{\mathsf{range}})$ or $y = x_{\mathsf{class}}$ by the axiomatic triple $(\mathsf{range}, \mathsf{range}, \mathsf{class})$. If $y = x_{\mathsf{class}}$ and $(u, v) \in Ext'(x_{\mathsf{range}})$, we have two cases: if $(u, v) \in \{(x_{\mathsf{s}}, x_{\mathsf{o}}) \mid (\mathsf{s}, \mathsf{range}, \mathsf{o}) \in Ax\}$ then by the construction of $\mathcal{I}'$ we have $v \in Class' = CExt'(x_{\mathsf{class}}) = CExt'(y)$; if $(u, v) \in Ext(x_{\mathsf{range}})$ then by the construction of $Class'$ we have $v \in Class'$. Now if $(x_{\mathsf{range}}, y) \in Ext(x_{\mathsf{range}})$, then by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and then because $v \in Res'$ we obtain $u \in CExt'(y)$.

- $x = x_{\mathsf{sp}}$: almost the same proof for x_{sp} in the previous condition works here considering v instead of u and changing x_{dom} with x_{range}, because, by the construction of $\mathcal{I}'$, $x_{\mathsf{member}} \in Prop'$ (axiomatic triple $(\mathsf{member}, \mathsf{dom}, \mathsf{res})$).

- $x = x_{\mathsf{sc}}$: almost the same proof for x_{sc} in the previous condition works here considering v instead of u and changing x_{dom} with x_{range}, because, by the construction of $\mathcal{I}'$, $x_{\mathsf{res}} \in Class'$ (axiomatic triple $(\mathsf{type}, \mathsf{dom}, \mathsf{res})$).

21

Then, in all cases $\mathcal{I}'$ satisfies this condition for G.

7. RDFS Additionals:

   (a) If $x \in Class'$ then by the construction of $\mathcal{I}'$ we have $(x, x_{\text{res}}) \in Ext'(x_{\text{sc}})$.

   (b) If $x \in CExt'(x_{\text{datatype}})$ then $x = x_{\text{xmlLit}}$ then, by the construction of $\mathcal{I}'$ and because $(\text{xmlLit}, \text{sc}, \text{literal})$ is an axiomatic triple, we have $(x, x_{\text{literal}}) \in Ext'(x_{\text{sc}})$.

   (c) If $x \in CExt'(x_{\text{contMP}})$ then $x = x_{\_\text{i}}$ for some i, and then by the construction of $Ext'(x_{\text{sp}})$ in $\mathcal{I}'$, we have that $(x, x_{\text{member}}) \in Ext'(x_{\text{sp}})$.

Now, what we have shown is that $\mathcal{I}' \models G$, and from $G \models H$ we obtain that $\mathcal{I}' \models H$. Note that if we restrict $\mathcal{I}'$ to vocabulary ρdf we obtain the initial interpretation $\mathcal{I}$ that satisfies all conditions that have to do with ρdf for H and then $\mathcal{I} \models_{\rho\text{df}} H$, and then $G \models_{\rho\text{df}} H$ completing the proof.

B.2 Proof of Proposition 1

Proof. It is not difficult to see that the triples in the set $A = \{(\text{sp}, \text{sp}, \text{sp}),$ $(\text{sc}, \text{sp}, \text{sc}), (\text{type}, \text{sp}, \text{type}), (\text{dom}, \text{sp}, \text{dom}), (\text{range}, \text{sp}, \text{range})\}$ an their existencial version replacing subject or predicat by blank nodes, are satisfied by every ρdf model, and then they are ρdf–entailed by every graph G. The rest of the proof follows by a analysis of cases, taking into account that, for a ground triple to be satisfied by every model, all its components must be elements in ρdf.

B.3 Proof of Theorem 2

Proof. $\Leftarrow$) Let $\mathcal{I} = (Res, Prop, Class, Ext, CExt, Lit, Int)$ be an ρdf model of G. By definition, $\mathcal{I}$ is also a reflexive–relaxed ρdf model for G and then form $G \models_{\rho\text{df}}^{\text{nrx}} H$ we obtain that $\mathcal{I}$ is also a reflexive–relaxed ρdf model for H. Now, $\mathcal{I}$ is an interpretation that satisifies the conditions of Definition 4 for H, and $\mathcal{I}$ is such that $Ext(Int(\text{sp}))$ and $Ext(Int(\text{sc}))$ are reflexive relations, then $\mathcal{I}$ satisfies all the conditions of Definition 3 for H and then $\mathcal{I}$ is a model under ρdf for H, completing this part of the proof.

$\Rightarrow$) Let $\mathcal{I} = (Res, Prop, Class, Ext, CExt, Lit, Int)$ be a reflexive-relaxed model of G, and let $\mathcal{I}'$ be the model obtained from $\mathcal{I}$ completing the relations $Ext(Int(\text{sp}))$ and $Ext(Int(\text{sc}))$ with the diagonals in $Prop$ and $Class$ respectively. Then $\mathcal{I}'$ is an RDFS model for G (satisfies all the conditions in Definition 6 for G), and then from $G \models H$ we have that $\mathcal{I}'$ is an RDFS model for H. Now we will show that $\mathcal{I}$ satisfies all triples $(s, p, o) \in H$. Let A be the extension function that $\mathcal{I}'$ use in modeling H, then we know that $(Int_A(s), Int_A(p)) \in Ext(Int(p))$ for every $(s, p, o) \in H$. We also know that $\mathcal{I}$ and $\mathcal{I}'$ differ only in the *diagonal* of $Ext(Int(\text{sp}))$ and $Ext(Int(\text{sc}))$. Now, because H does not contain triples of the form (x, sp, x) nor (x, sc, x) nor their existential versions replacing subject or object for a blank node, the same extension function A is such that in $\mathcal{I}$ $(Int_A(s), Int_A(p)) \in Ext(Int(p))$ for every $(s, p, o) \in H$ and then satisfies all the conditions of Definition 4, and finally $\mathcal{I}$ is a reflexive–relaxed model for H, completing this part of the proof.

22

B.4 Proof of Proposition 1

Proof. It is evident that, because the interpretation of $\mathtt{sp}$ is not necessary reflexive over property names, non of the triples in $A = \{(\mathtt{sp}, \mathtt{sp}, \mathtt{sp}), (\mathtt{sc}, \mathtt{sp}, \mathtt{sc}),$ $(\mathtt{type}, \mathtt{sp}, \mathtt{type}), (\mathtt{dom}, \mathtt{sp}, \mathtt{dom}), (\mathtt{range}, \mathtt{sp}, \mathtt{range})\}$ are axiomatic for $\models_{\rho\mathrm{df}}^{\mathrm{nrx}}$. Finally, the fact that $\models_{\rho\mathrm{df}}^{\mathrm{nrx}} \subseteq \models_{\rho\mathrm{df}}$ complete the proof.

B.5 Proof of Theorem 3

First, in the definition of ρdf models for RDF graphs (Definition 3), the only condition that has to do with the graph being modeled is condition 1 (*Simple*). The other conditions have to do only with the interpretation itself. All this implies that in testing if an interpretation $\mathcal{I}$ that is an ρdf model for a graph G, is also an ρdf model for a graph H, we only have to test if $\mathcal{I}$ satisfies condition 1 for H, because $\mathcal{I}$ already satisfies all other conditions (it is already an ρdf model for G).

We split the proof of Theorem 3 in two parts. We first prove the following lemma stating the soundness of the set of rules for $\models_{\rho\mathrm{df}}$.

Lemma 5. *Let G and H be graphs that do not mention RDFS vocabulary outside ρdf. Then if $H \to G$, or $H \subseteq G$, or if there is an instantiation $\frac{R}{R'}$ of a rule 2–7 such that $R \subseteq G$ and $H = G \cup R'$, then $G \models_{\rho\mathrm{df}} H$.*

Proof. Let $\mathcal{I} = (Res, Prop, Class, Ext, CExt, Int)$ be an interpretation such that $\mathcal{I} \models_{\rho\mathrm{df}} G$, i.e. $\mathcal{I}$ satisfies all the conditions in Definition 3. We know that $\mathcal{I}$ satisfies condition 1 for G and then let $A : \mathbf{B} \to Res$ be a function such that $Int(p) \in Prop$ and $(Int_A(s), Int_A(o)) \in Ext(Int(p))$ for every triple $(s, p, o) \in G$, We split the proof in cases for every set of rules from 1 to 7.

1. Simple:
   (a) We must show that if $H \to G$ then $G \models_{\rho\mathrm{df}} H$. Let μ be a map such that $\mu(H) \subseteq G$. Consider the function $A' : \mathbf{B} \to Res$ defined as

   $$A'(x) = \begin{cases} A(\mu(x)) & \text{if } \mu(x) \in \mathbf{B} \\ Int(\mu(x)) & \text{if } \mu(x) \notin \mathbf{B} \end{cases}$$

   Note that: (1) if $x \in \mathbf{B}$ and $\mu(x) \in \mathbf{B}$ then $Int_A(\mu(x)) = A(\mu(x)) = A'(x) = Int_{A'}(x)$, (2) If $x \in \mathbf{B}$ but $\mu(x) \notin \mathbf{B}$ then $Int_A(\mu(x)) = Int(\mu(x)) = A'(x) = Int_{A'}(x)$, and (3) if $x \notin \mathbf{B}$ then $\mu(x) = x$ and $Int_A(\mu(x)) = Int(x) = Int_{A'}(x)$. We have then that for all $x \in \mathbf{UB}$, $Int_A(\mu(x)) = Int_{A'}(x)$. Let $(s, p, o) \in H$, then $(\mu(s), \mu(p), \mu(o)) = (\mu(s), p, \mu(o)) \in G$. By $\mathcal{I} \models_{\rho\mathrm{df}} G$ we have that $Int(p) \in Prop$ and $(Int_A(\mu(s)), Int_A(\mu(o))) \in Ext(Int(p))$, and finally $(Int_{A'}(s), Int_{A'}(o)) \in Ext(Int(p))$, obtaining $\mathcal{I}$ satisfies condition 1 of Defintion 3 for H (with function A') and also satisfies all other conditions of Definition 3, and then $\mathcal{I} \models_{\rho\mathrm{df}} H$.
   (b) Note that if $H \subseteq G$, then $H \to G$ and then $G \models_{\rho\mathrm{df}} H$.

2. Subproperty:
   (a) Let $(a, \mathtt{sp}, b), (b, \mathtt{sp}, c) \in G$, then $(Int_A(a), Int_A(b)) \in Ext(Int(\mathtt{sp}))$ and $(Int_A(b), Int_A(c)) \in Ext(Int(\mathtt{sp}))$. Now because $\mathcal{I}$ satisfies condition 4 we have that $Int_A(a), Int_A(c) \in Prop$ and then by transitivity $(Int_A(a), Int_A(c)) \in Ext(Int(\mathtt{sp}))$, then $\mathcal{I}$ satisfies condition 1 for $G \cup \{(a, \mathtt{sp}, c)\} = H$ and then $\mathcal{I} \models_{\rho df} H$.
   (b) Let $(a, \mathtt{sp}, b), (x, a, y) \in G$. First note that we need that $a, b \in \mathbf{U}$ for this rule to be applicable. We have that $Int(a) \in Prop$ and $(Int_A(x), Int_A(y)) \in Ext(Int(a))$, and $(Int(a), Int(b)) \in Ext(Int(\mathtt{sp}))$. By condition 4, $Int(b) \in Prop$ and $Ext(Int(a)) \subseteq Ext(Int(b))$ and then $(Int_A(x), Int_A(y)) \in Ext(Int(b))$, then we have that $\mathcal{I}$ satisfies condition 1 for $G \cup \{(x, b, y)\} = H$ and then $\mathcal{I} \models_{\rho df} H$.
3. Subclass:
   (a) The same proof for rule 2a works changing $\mathtt{sp}$ by $\mathtt{sc}$ and $Prop$ by $Class$.
   (b) Let $(a, \mathtt{sc}, b), (x, \mathtt{type}, a) \in G$, then $(Int_A(x), Int_A(a)) \in Ext(Int(\mathtt{type}))$, and $(Int_A(a), Int_A(b)) \in Ext(Int(\mathtt{sc}))$. By condition 6 we have $Int_A(a) \in Class$ and $Int_A(x) \in CExt(Int_A(a))$. By condition 5, $Int_A(b) \in Class$ and $CExt(Int(a)) \subseteq CExt(Int(b))$, then $Int_A(x) \in CExt(Int_A(b))$, and then by condition 6 $(Int_A(x), Int_A(b)) \in Ext(Int(\mathtt{type}))$. We have that $\mathcal{I}$ satisfies condition 1 for $G \cup \{(x, \mathtt{type}, b)\} = H$ and then $\mathcal{I} \models_{\rho df} H$.
4. Typing:
   (a) Let $(a, \mathtt{dom}, b), (x, a, y) \in G$. First note that we need that $a \in \mathbf{U}$ for this rule to be applicable. Now, we have that (1) $(Int(a), Int_A(b)) \in Ext(Int(\mathtt{dom}))$, and (2) $Int(a) \in Prop$ and $(Int_A(x), Int_A(y)) \in Ext(Int(a))$. From condition 6 we obtain $Int_A(x) \in CExt(Int_A(b))$, and applying condition 6 again we have that $(Int_A(x), Int_A(b)) \in Ext(Int(\mathtt{type}))$ then $\mathcal{I}$ satisfies condition 1 for $G \cup \{(x, \mathtt{type}, b)\} = H$ and then $\mathcal{I} \models_{\rho df} H$.
   (b) The same proof for rule 4a works changing $\mathtt{dom}$ by $\mathtt{range}$ and x by y.
5. Implicit Typing:
   (a) Let $(a, \mathtt{dom}, b), (c, \mathtt{sp}, a), (x, c, y) \in G$. First note that we need that $c \in \mathbf{U}$ for this rule to be applicable. Now, we have that (1) $(Int_A(a), Int_A(b)) \in Ext(Int(\mathtt{dom}))$, (2) $(Int(c), Int_A(a)) \in Ext(Int(\mathtt{sp}))$, and (3) $Int(c) \in Prop$ and $(Int_A(x), Int_A(y)) \in Ext(Int(c))$. From (2) by condition 4 we have $Int(c), Int_A(a) \in Prop$ and $Ext(Int(c)) \subseteq Ext(Int_A(a))$, and then from (3) we obtain that $(Int_A(x), Int_A(y)) \in Ext(Int_A(a))$. From this last result, (1), and condition 6 we obtain that $Int_A(b) \in Class$ and $Int_A(x) \in CExt(Int_A(b))$. Finally applying condition 6 again we have that $(Int_A(x), Int_A(b)) \in Ext(Int(\mathtt{type}))$ then $\mathcal{I}$ satisfies condition 1 for $G \cup \{(x, \mathtt{type}, b)\} = H$ and then $\mathcal{I} \models_{\rho df} H$.
   (b) The same proof for rule 5a works changing $\mathtt{dom}$ by $\mathtt{range}$ and x by y.
6. Subproperty Reflexivity:
   (a) Let $(x, a, y) \in G$. First note that we need that $a \in \mathbf{U}$ for this rule to be applicable. We have that $Int(a) \in Prop$ and then by the reflexivity of $Ext(Int(\mathtt{sp}))$ over $Prop$, we obtain that $(Int(a), Int(a)) \in Ext(Int(\mathtt{sp}))$, then we have that $\mathcal{I}$ satisfies condition 1 for $G \cup \{(a, \mathtt{sp}, a)\} = H$ and then $\mathcal{I} \models_{\rho df} H$.

(b) Let $(a, \mathtt{sp}, b) \in G$. By condition 4 we have that $Int(a), Int(b) \in Prop$ and the proof follows the same argument as for rule 6a.

(c) The triples $(p, \mathtt{sp}, p)$ with $p \in \rho\mathrm{df}$ are satisfied by any interpretation (see Proposition 1) then $\mathcal{I} \models_{\rho\mathrm{df}} G \cup \{(p, \mathtt{sp}, p)\}$ for every $p \in \rho\mathrm{df}$.

(d) Let $(a, p, x) \in G$ with $p \in \{\mathtt{dom}, \mathtt{range}\}$. By the new conditions of $\rho\mathrm{df}$ models, we have that $Int(a) \in Prop$ and the proof follows the same argument as for rule 6a.

7. Subclass Reflexivity:

(a) Let $(a, \mathtt{sc}, b) \in G$. By condition 5 we have that $Int(a), Int(b) \in Class$ and then by the reflexivity of $Ext(Int(\mathtt{sc}))$ over $Class$ we obtain that $(Int_A(a), Int_A(a)), (Int_A(b), Int_A(b)) \in Ext(Int(\mathtt{sc}))$, then we have that $\mathcal{I}$ satisfies condition 1 for $G \cup \{(a, \mathtt{sc}, a), (b, \mathtt{sc}, b)\} = H$ and then $\mathcal{I} \models_{\rho\mathrm{df}} H$.

(b) Let $(x, p, a) \in G$ with $p \in \{\mathtt{dom}, \mathtt{range}, \mathtt{type}\}$. By the new condition of $\rho\mathrm{df}$ models, we have that $Int(a) \in Class$ and the proof follows the same argument as for rule 7a.

Finally because we choose an arbitrary model $\mathcal{I}$ we have that $G \models_{\rho\mathrm{df}} H$.

Lemma 6. *Let G and H be graphs that do not mention RDFS vocabulary outside $\rho\mathrm{df}$. Then if $H \to G$, or $H \subseteq G$, or if there is an instantiation $\frac{R}{R'}$ of a rule 2–5 such that $R \subseteq G$ and $H = G \cup R'$, then $G \models_{\rho\mathrm{df}}^{\mathrm{nrx}} H$.*

Proof. Follows from the simple observation that in the proof of Lemma 5 the condition of reflexivity of the interpretations of $\mathtt{sp}$ and $\mathtt{sc}$ are necessary only for rules 6 and 7.

To state the completeness of the set of rules, we must introduce the following notion of $\rho\mathrm{df}$ *closure* of a graph. Define the graph $\rho\mathrm{df\text{-}cl}(G)$ as the *closure* of G under the application of rules 2 to 7. Note that $\rho\mathrm{df\text{-}cl}(G)$ is an RDF graph over $\mathrm{universe}(G) \cup \rho\mathrm{df}$, that is a superset of G, and that is obtained after a finite number of application of rules.

Lemma 7. *Given a graph G that do not mention RDFS vocabulary outside $\rho\mathrm{df}$, define the interpretation $\mathcal{I}_G = (Res, Prop, Class, Ext, CExt, Lit, Int)$ such that:*

- $Res = \mathrm{universe}(G) \cup \rho\mathrm{df}$.
- $Prop = \{p \in \mathrm{voc}(G) \mid (s, p, o) \in \rho\mathrm{df\text{-}cl}(G)\} \cup \rho\mathrm{df} \cup$ $\{p \in \mathrm{universe}(G) \mid (p, \mathtt{sp}, x), (y, \mathtt{sp}, p), (p, \mathtt{dom}, z), \text{ or } (p, \mathtt{range}, v) \in G\}$.
- $Class = \{c \in \mathrm{universe}(G) \mid (x, \mathtt{type}, c) \in G\} \cup$ $\{c \in \mathrm{universe}(G) \mid (c, \mathtt{sc}, x), (y, \mathtt{sc}, c), (z, \mathtt{dom}, c), \text{ or } (v, \mathtt{range}, c) \in G\}$.
- $Ext : Prop \to 2^{Res \times Res}$ the extension function such that:
  - if $p \in \mathbf{U} \cap Prop$ then $Ext(p) = \{(s, o) \mid (s, p, o) \in \rho\mathrm{df\text{-}cl}(G)\}$
  - if $p \in \mathbf{B} \cap Prop$ then $Ext(p) = \{(s, o) \mid (p', \mathtt{sp}, p), (s, p', o) \in \rho\mathrm{df\text{-}cl}(G)\}$.
- $CExt : Class \to 2^{Res}$ a function such that $CExt(c) = \{x \in \mathrm{universe}(G) \mid (x, \mathtt{type}, c) \in \rho\mathrm{df\text{-}cl}(G)\}$.
- $Lit = \mathrm{universe}(G) \cap \mathbf{L}$.
- Int the identity function over $\mathrm{universe}(G) \cup \rho\mathrm{df}$.

Then for every RDF graph G, we have that $\mathcal{I}_G \models_{\rho\mathrm{df}} G$.

Proof. We must show that $\mathcal{I}_G$ satisfies all the conditions of Definition 3 for G.

1. Simple:
   (a) First note that by construction of $\rho\mathrm{df\text{-}cl}(G)$, $Res = \mathrm{universe}(\rho\mathrm{df\text{-}cl}(G)) = \mathrm{universe}(G) \cup \rho\mathrm{df}$. Take the function $A : \mathbf{B} \to \mathrm{universe}(G) \cup \rho\mathrm{df}$ such that its restriction to the set of blanks nodes of G results in the identity function. Now let $(s, p, o) \in G$, then $p \in \mathbf{U}$ and $Int(p) = p \in Prop$ by construction of $Prop$ because $G \subseteq \rho\mathrm{df\text{-}cl}(G)$. We also have that $(Int_A(s), Int_A(o)) = (s, o) \in Ext(Int(p)) = Ext(p)$ by the definition of Ext because $G \subseteq \rho\mathrm{df\text{-}cl}(G)$. Finally we have that $\mathcal{I}_G$ satisfies condition 1 for G.

2. Subproperty:
   (a) Let $(a, b), (b, c) \in Ext(Int(\mathtt{sp})) = Ext(\mathtt{sp})$, then by construction of $\mathcal{I}_G$ (because $\mathtt{sp} \notin \mathbf{B}$) we have that $(a, \mathtt{sp}, b)$, $(b, \mathtt{sp}, c) \in \rho\mathrm{df\text{-}cl}(G)$, then $a, b, c \in Prop$. Because $\rho\mathrm{df\text{-}cl}(G)$ is closed under application of rule 2a, we have that $(a, \mathtt{sp}, c) \in \rho\mathrm{df\text{-}cl}(G)$ and then $(a, c) \in Ext(\mathtt{sp}) = Ext(Int(\mathtt{sp}))$. We conclude that $Ext(Int(\mathtt{sp}))$ is a transitive relation. We must show that $Ext(Int(\mathtt{sp}))$ is also reflexive over $Prop$. Let $a \in Prop$, by the definition of $Prop$ we have three cases: (1) $(x, a, y) \in \rho\mathrm{df\text{-}cl}(G)$; (2) $a \in \rho\mathrm{df}$; (3) $(a, \mathtt{sp}, b)$, $(b, \mathtt{sp}, a)$, $(a, \mathtt{dom}, x)$, or $(a, \mathtt{range}, x) \in \rho\mathrm{df\text{-}cl}(G)$. Because $\rho\mathrm{df\text{-}cl}(G)$ is closed under application of rules 6 we obtain that in any case $(a, \mathtt{sp}, a) \in \rho\mathrm{df\text{-}cl}(G)$ and then $(a, a) \in Ext(\mathtt{sp}) = Ext(Int(\mathtt{sp}))$ and then $Ext(Int(\mathtt{sp}))$ is reflexive over $Prop$.
   (b) Let $(a, b) \in Ext(Int(\mathtt{sp})) = Ext(\mathtt{sp})$, then by construction of $\mathcal{I}_G$ we have that $(a, \mathtt{sp}, b) \in \rho\mathrm{df\text{-}cl}(G)$, and we also have that $a, b \in Prop$. We must show that $Ext(a) \subseteq Ext(b)$. If $Ext(a) = \emptyset$ the condition holds. Suppose then that $(x, y) \in Ext(a)$, we have two cases:
      − if $a \in \mathbf{U}$, by definition $(x, a, y) \in \rho\mathrm{df\text{-}cl}(G)$. Now, if $b \in \mathbf{U}$ because $\rho\mathrm{df\text{-}cl}(G)$ is closed under application of rule 2b we have that $(x, b, y) \in \rho\mathrm{df\text{-}cl}(G)$ and then $(x, y) \in Ext(b)$. If $b \in \mathbf{B}$ then because $(a, \mathtt{sp}, b), (x, a, y) \in \rho\mathrm{df\text{-}cl}(G)$ by the construction of $\mathcal{I}_G$ we have that $(x, y) \in Ext(b)$.
      − if $a \in \mathbf{B}$, by the construction of $\mathcal{I}_G$ there exists a' such that $(a', \mathtt{sp}, a)$, $(x, a', y) \in \rho\mathrm{df\text{-}cl}(G)$. Note that because $\rho\mathrm{df\text{-}cl}(G)$ is closed under application of rule 2a, we have $(a', \mathtt{sp}, b) \in \rho\mathrm{df\text{-}cl}(G)$. Now, if $b \in \mathbf{U}$ because $(a', \mathtt{sp}, b), (x, a', y) \in \rho\mathrm{df\text{-}cl}(G)$ and $\rho\mathrm{df\text{-}cl}(G)$ is closed under application of rule 2b we have that $(x, b, y) \in \rho\mathrm{df\text{-}cl}(G)$ and then $(x, y) \in Ext(b)$. If $b \in \mathbf{B}$ then because $(a', \mathtt{sp}, b), (x, a', y) \in \rho\mathrm{df\text{-}cl}(G)$ by the construction of $\mathcal{I}_G$ we have that $(x, y) \in Ext(b)$.
      We have shown that in any case $(x, y) \in Ext(b)$ and then $Ext(a) \subseteq Ext(b)$.

3. Subclass:
   (a) Let $(a, b), (b, c) \in Ext(Int(\mathtt{sc})) = Ext(\mathtt{sc})$, then by the construction of $\mathcal{I}_G$ we have that $(a, \mathtt{sc}, b)$, $(b, \mathtt{sc}, c) \in \rho\mathrm{df\text{-}cl}(G)$, then $a, b, c \in Class$. Because $\rho\mathrm{df\text{-}cl}(G)$ is closed under application of rule 3a, we have that

$(a, \mathtt{sc}, c) \in \rho\text{df-cl}(G)$ and then $(a, c) \in Ext(\mathtt{sc}) = Ext(Int(\mathtt{sc}))$. We conclude that $Ext(Int(\mathtt{sc}))$ is a transitive relation. We must show that $Ext(Int(\mathtt{sc}))$ is also reflexive over $Class$. Let $a \in Class$, by the definition of $Class$ we have two cases: (1) $(x, \mathtt{type}, a) \in \rho\text{df-cl}(G)$; (2) $(a, \mathtt{sc}, b)$, $(b, \mathtt{sc}, a)$, $(x, \mathtt{dom}, a)$, or $(x, \mathtt{range}, a) \in \rho\text{df-cl}(G)$. Because $\rho\text{df-cl}(G)$ is closed under application of rules 7 we obtain that in any case $(a, \mathtt{sc}, a) \in \rho\text{df-cl}(G)$ and then $(a, a) \in Ext(\mathtt{sc}) = Ext(Int(\mathtt{sc}))$ and then $Ext(Int(\mathtt{sc}))$ is reflexive over $Class$.

(b) Let $(a, b) \in Ext(Int(\mathtt{sc})) = Ext(\mathtt{sc})$, then by the construction of $\mathcal{I}_G$ we have that $(a, \mathtt{sc}, b) \in \rho\text{df-cl}(G)$, and we also have that $a, b \in Class$. We must show that $CExt(a) \subseteq CExt(b)$. If $CExt(a) = \emptyset$ the property holds. Suppose that $x \in CExt(a)$, then by definition $(x, \mathtt{type}, a) \in \rho\text{df-cl}(G)$. Now, because $\rho\text{df-cl}(G)$ is closed under application of rule 3b we have that $(x, \mathtt{type}, b) \in \rho\text{df-cl}(G)$ and then by the construction of $\mathcal{I}_G$ we have $x \in CExt(b)$.

4. Typing I:

(a) Let $(x, a) \in Ext(Int(\mathtt{type})) = Ext(\mathtt{type})$, then by the construction of $\mathcal{I}_G$ we have that $a \in Class$ and $(x, \mathtt{type}, a) \in \rho\text{df-cl}(G)$, and then by construction of $CExt(a)$ we have that $x \in CExt(a)$. Suppose now that $a \in Class$ and $x \in CExt(a)$ then by construction of $CExt(a)$ we have that $(x, \mathtt{type}, a) \in \rho\text{df-cl}(G)$ and then $(x, a) \in Ext(\mathtt{type}) = Ext(Int(\mathtt{type}))$. We have shown that $(x, a) \in Ext(Int(\mathtt{type}))$ iff $x \in CExt(a)$.

(b) Suppose that $(a, b) \in Ext(Int(\mathtt{dom})) = Ext(\mathtt{dom})$ and $(x, y) \in Ext(a)$, we must show that $x \in CExt(b)$. First, by the construction of $\mathcal{I}_G$, $(a, \mathtt{dom}, b) \in \rho\text{df-cl}(G)$, we have two cases:
  - if $a \in \mathbf{U}$ then by the construction of $\mathcal{I}_G$, $(x, a, y) \in \rho\text{df-cl}(G)$ and because $\rho\text{df-cl}(G)$ is closed under application of rule 4a we have that $(x, \mathtt{type}, b) \in \rho\text{df-cl}(G)$, and then by construction of $CExt(b)$, $x \in CExt(b)$.
  - if $a \in \mathbf{B}$, because $(x, y) \in Ext(a)$, by construction of $\mathcal{I}_G$ there exists a' such that $(a', \mathtt{sp}, a), (x, a', y) \in \rho\text{df-cl}(G)$, and because $\rho\text{df-cl}(G)$ is closed under application of rule 5a we have that $(x, \mathtt{type}, b) \in \rho\text{df-cl}(G)$, and then by construction of $CExt(b)$, $x \in CExt(b)$.

  We have shown that in any case $x \in CExt(b)$.

(c) Suppose that $(a, b) \in Ext(Int(\mathtt{range})) = Ext(\mathtt{range})$ and $(x, y) \in Ext(a)$, we must show that $y \in CExt(b)$. First, by construction of $\mathcal{I}_G$, $(a, \mathtt{dom}, b) \in \rho\text{df-cl}(G)$, we have two cases:
  - if $a \in \mathbf{U}$ then $(x, a, y) \in \rho\text{df-cl}(G)$ and because $\rho\text{df-cl}(G)$ is closed under application of rule 4b we have that $(y, \mathtt{type}, b) \in \rho\text{df-cl}(G)$, and then by construction of $CExt(b)$, $y \in CExt(b)$.
  - if $a \in \mathbf{B}$, because $(x, y) \in Ext(a)$, by construction of $\mathcal{I}_G$ there exists a' such that $(a', \mathtt{sp}, a), (x, a', y) \in \rho\text{df-cl}(G)$, and because $\rho\text{df-cl}(G)$ is closed under application of rule 5b we have that $(y, \mathtt{type}, b) \in \rho\text{df-cl}(G)$, and then by construction of $CExt(b)$, $y \in CExt(b)$.

  We have shown that in any case $y \in CExt(b)$.

5. Typing II: all this condition hold by definition of *Prop* and *Class*.

We have shown that $\mathcal{I}_G$, satisfies all the conditions of Definition 3 for G, and then $\mathcal{I}_G \models_{\rho\mathrm{df}} G$.

Similarly as we define $\rho\mathrm{df}\text{-}\mathrm{cl}(G)$, define $\mathrm{nrx}\text{-}\rho\mathrm{df}\text{-}\mathrm{cl}(G)$ but using only rules from 2 to 5. Then we have the following Lemma.

Lemma 8. *For a graph G, consider the interpretation $\mathcal{I}_G$ as in Lemma 7 but using* $\mathrm{nrx}\text{-}\rho\mathrm{df}\text{-}\mathrm{cl}(G)$ *instead of* $\rho\mathrm{df}\text{-}\mathrm{cl}(G)$. *Then* $\mathcal{I}_G \models_{\rho\mathrm{df}}^{\mathrm{nrx}} G$.

Proof. Follows from the simple observation that in the proof of Lemma 7 rules 6 and 7 are neede only to show the reflexivity of the interepretations of $\mathtt{sp}$ and of $\mathtt{sc}$.

Lemma 9. *Let G, H be RDF graphs that do not mention RDFS vocabulary outside $\rho\mathrm{df}$. If $G \models_{\rho\mathrm{df}} H$ then there is a map $H \to \rho\mathrm{df}\text{-}\mathrm{cl}(G)$. If H does not conatin triples of the form $(x, \mathtt{sp}, x)$ nor $(x, \mathtt{sc}, x)$ then, if $G \models_{\rho\mathrm{df}}^{\mathrm{nrx}} H$ then there is a map $H \to \mathrm{nrx}\text{-}\rho\mathrm{df}\text{-}\mathrm{cl}(G)$.*

Proof. First for $\models_{\rho\mathrm{df}}$. Consider the interpretation,

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

as defined in Lemma 7, then we have that $\mathcal{I}_G \models_{\rho\mathrm{df}} G$ and because $G \models_{\rho\mathrm{df}} H$ we have $\mathcal{I}_G \models_{\rho\mathrm{df}} H$. Then we know that $\mathcal{I}_G$ satisfies condition 1 (*Simple*) for H, and then, there exists a function $A : B \to \mathrm{universe}(G) \cup \rho\mathrm{df}$ such that for each $(s, p, o) \in H$, $Int(p) \in Prop$ and $(Int_A(s), Int_A(o)) \in Ext(Int(p))$. Now because $p \in \mathbf{U}$ (p is the predicate in a triple in H), we know that $Int(p) = Int_A(p) = p$, and $Ext(Int(p)) = Ext(p) = \{(s, o) \mid (s, p, o) \in \rho\mathrm{df}\text{-}\mathrm{cl}(G)\}$. Finally, because $(Int_A(s), Int_A(o)) \in Ext(Int(p))$ we have that, for each $(s, p, o) \in H$, $(Int_A(s), Int_A(p), Int_A(o)) \in \rho\mathrm{df}\text{-}\mathrm{cl}(G)$ then $Int_A : H \to \rho\mathrm{df}\text{-}\mathrm{cl}(G)$ is a map such that $Int_A(H) \subseteq \rho\mathrm{df}\text{-}\mathrm{cl}(G)$, that is a map $H \to \rho\mathrm{df}\text{-}\mathrm{cl}(G)$.

For $\models_{\rho\mathrm{df}}^{\mathrm{nrx}}$, consider $\mathcal{I}_G$ as defined in Lemma 8. The proof follows the same argument as above, but considering the fact that H does not contain triples of the form $(x, \mathtt{sc}, x)$ nor $(x, \mathtt{sp}, x)$.

The proof of Theorem 3 follows directly from Lemmas 5 and 9.

B.6   Proof of Corollary 2

The proof of Corollary 2 follows directly from Lemmas 6 and 9.

C   Proofs of Section 4

Throughout this section, X stands for one of the fragments of $\rho\mathrm{df}$ listed in Figure 1. By a *non X-rule* we will understand a rule which mention RDFS vocabulary outside X.

Let r be one of the rules listed in Section 3.1. We write $P \vdash_r P'$ if one of the following cases hold:

- r is (1a) and there is a map $\mu : P' \to P$; or
- r is (1b) and $P' \subseteq P$; or
- r is one of the rules (2)-(7) and there is an instance $\frac{R}{R'}$ of r with $R \subseteq P$ and $P' = P \cup R'$.

C.1 Proof of Theorem 4

The proof follows directly from Lemma 9.

C.2 Proof of Theorem 5

Lemma 10. *Let X a fragments of ρdf. If G is an X-graph and we have a proof of H from G using only a single step of rule (1), then H has at most RDFS vocabulary in X*

Proof. It suffices to analize the application of these rules:

1. If the rule applied is (1b), that is, $H_2 \subseteq G$. Hence $\operatorname{voc} H \subseteq \operatorname{voc} G$. By hypotheses, we get that H is an X-graph.
2. If the rule applied is (1a), that is, there is a map $\mu : H_2 \to G$. For any $(X, Y, Z) \in H_2$ we have $(\mu(X), \mu(Y), \mu(Z)) \in G$, that is, the RDFS vocabulary appearing among $\mu(X), \mu(Y), \mu(Z)$ is in X. Hence the same can be said about each of X, Y and Z, because they may be blanks, and so it is not an RDFS vocabulary outside X, or they are not blanks, and so μ acts as the identity over them, which implies that they actually are equal to their image under μ, which are not in RDFS vocabulary outside X. Thus H is an X-graph.

We will say that a the application of a rule r *adds* a triple t to a graph G if $t \notin G$ and $G \vdash_r (G \cup \{t\} \cup H)$ for some graph H.

Lemma 11. *Assume that X is a fragment of ρdf and that G is an X-graph. Fix $a \in \operatorname{voc}(G)$. Let t_{sp} be the triple (A, sp, A) and let t_{sc} be the triple (A, sc, A). Suppose that neither t_{sp} nor t_{sp} belongs to G.*
  Then:

- *The application of rules (2a), (3), (6b), (6c), (6d) and (7a), add to $G \cup \{t_{\mathsf{sc}}\}$ and to $G \cup \{t_{\mathsf{sp}}\}$ the same triples that they add to G; they are $\operatorname{voc}(G) \cup \{\mathsf{sc}, \mathsf{sp}\}$-rules.*
- *The application of rule (2b) adds a triple (A, B, A) to $G \cup \{t_{\mathsf{sc}}\}$ but not to G only if $(\mathsf{sc}, \mathsf{sp}, B) \in G$, it adds (A, B, A) to $G \cup \{t_{\mathsf{sp}}\}$ but not to G only if $(\mathsf{sp}, \mathsf{sp}, B) \in G$, and otherwise it adds the same triples to $G \cup \{t_{\mathsf{sc}}\}$, $G \cup \{t_{\mathsf{sp}}\}$, and G. In any case, this rule is a $\operatorname{voc}(G) \cup \{\mathsf{sc}, \mathsf{sp}\}$-rule.*
- *The application of rule (4a) adds a triple (A, type, B) to $G \cup \{t_{\mathsf{sc}}\}$ but not to G only if $(\mathsf{sc}, \mathsf{dom}, B) \in G$, it adds (A, type, B) to $G \cup \{t_{\mathsf{sp}}\}$ but not to G only if $(\mathsf{sp}, \mathsf{dom}, B) \in G$, and it cannot be applied otherwise. If (4a) were applied, it will be a $\{\mathsf{type}\} \cup (\operatorname{voc}(G) \cup \{\mathsf{sc}, \mathsf{sp}\})$-rule.*

- *The rule (4b) has the same behavior than rule (4a), but with* `dom` *replaced by* `range`.
- *The application of rule (5a) adds a triple $(X, \mathtt{type}, A)$ to $G \cup \{t_{\mathtt{sp}}\}$ but not to G only if $\{(A, \mathtt{dom}, B), (X, A, Y)\} \subseteq G$, it adds $(A, \mathtt{type}, B)$ to $G \cup \{t_p\}$, for $p = \mathtt{sc}, \mathtt{sp}$, but not to G only if $\{(C, \mathtt{dom}, B), (p, \mathtt{sp}, C)\} \subseteq G$, and it cannot be applied otherwise. If (5a) were applied, it will be a $\{\mathtt{type}\} \cup (\mathrm{voc}(G) \cup \{\mathtt{sc}, \mathtt{sp}\})$-rule.*
- *The rule (5b) has the same behavior than rule (5b), but with* `dom` *replaced by* `range`.
- *The application of rule (6a) always adds $(p, \mathtt{sp}, p)$, for $p = \mathtt{sc}, \mathtt{sp}$, to $G \cup \{\mathtt{sp}\}$, which can be obtained directly by the application of rule (6c) for $p = \mathtt{sc}, \mathtt{sp}$.*

Moreover, if we assume that there is no triples in G with `sc`*, nor* `sp` *in the subject, then only rules (5) add triples to $G \cup \{t_{\mathtt{sc}}\}$ and to $G \cup \{t_{\mathtt{sp}}\}$, but the added triples are the same than rules (4) add to G.*

Proof. We only need to check some cases, and the statements of the lemma follows directly:

- For rule (2b), we have:
  1. The trivial application gives the instance $\dfrac{(A, \mathtt{sp}, A), (X, A, Y)}{(X, A, Y)}$.
  2. Assuming $(\mathtt{sc}, \mathtt{sp}, B) \in G$, we have the instance $\dfrac{(\mathtt{sc}, \mathtt{sp}, B), (A, \mathtt{sc}, A)}{(A, B, A)}$.
  3. Assuming $(\mathtt{sp}, \mathtt{sp}, B) \in G$, we have the instance $\dfrac{(\mathtt{sp}, \mathtt{sp}, B), (A, \mathtt{sp}, A)}{(A, B, A)}$.
- For rule (4a), we have
  1. Assuming $(\mathtt{sc}, \mathtt{dom}, B) \in G$, we have the instance $\dfrac{(\mathtt{sc}, \mathtt{dom}, B), (A, \mathtt{sc}, A)}{(A, \mathtt{type}, B)}$
  2. Assuming $(\mathtt{sp}, \mathtt{dom}, B) \in G$, we have the instance $\dfrac{(\mathtt{sp}, \mathtt{dom}, B), (A, \mathtt{sp}, A)}{(A, \mathtt{type}, B)}$
- For rule (4b), we have
  1. Assuming $(\mathtt{sc}, \mathtt{range}, B) \in G$, we have the instance $\dfrac{(\mathtt{sc}, \mathtt{range}, B), (A, \mathtt{sc}, A)}{(A, \mathtt{type}, B)}$
  2. Assuming $(\mathtt{sp}, \mathtt{range}, B) \in G$, we have the instance $\dfrac{(\mathtt{sp}, \mathtt{range}, B), (A, \mathtt{sp}, A)}{(A, \mathtt{type}, B)}$
- For rule (5a), we have
  1. Assuming $\{(A, \mathtt{dom}, B), (X, A, Y)\} \subseteq G$, we have the instance $\dfrac{(A, \mathtt{dom}, B), (A, \mathtt{sp}, A), (X, A, Y)}{(X, \mathtt{type}, B)}$, which has the same effect than the instance of rule (4a) $\dfrac{(A, \mathtt{dom}, B), (X, A, Y)}{(X, \mathtt{type}, B)}$, which can be directly applied to G.
  2. Assuming $\{(C, \mathtt{dom}, B), (\mathtt{sc}, \mathtt{sp}, A)\} \subseteq G$, we have the instance $\dfrac{(C, \mathtt{dom}, B), (\mathtt{sc}, \mathtt{sp}, A), (A, \mathtt{sc}, A)}{(A, \mathtt{type}, B)}$

3. Assuming $\{(C, \mathtt{dom}, B), (\mathtt{sp}, \mathtt{sp}, A)\} \subseteq G$, we have the instance

$$\frac{(C, \mathtt{dom}, B), (\mathtt{sp}, \mathtt{sp}, A), (A, \mathtt{sp}, A)}{(A, \mathtt{type}, B)}$$

– For rule (5b), we have

1. Assuming $\{(A, \mathtt{range}, B), (X, A, Y)\} \subseteq G$, we have the instance $\dfrac{(A, \mathtt{range}, B), (A, \mathtt{sp}, A), (X, A, Y)}{(Y, \mathtt{type}, B)}$, which has the same effect than the instance of rule (4a) $\dfrac{(A, \mathtt{range}, B), (X, A, Y)}{(Y, \mathtt{type}, B)}$, which can be directly applied to G.

2. Assuming $\{(C, \mathtt{range}, B), (\mathtt{sc}, \mathtt{sp}, A)\} \subseteq G$, we have the instance

$$\frac{(C, \mathtt{range}, B), (\mathtt{sc}, \mathtt{sp}, C), (A, \mathtt{sc}, A)}{(A, \mathtt{type}, B)}$$

3. Assuming $\{(C, \mathtt{range}, B), (\mathtt{sp}, \mathtt{sp}, A)\} \subseteq G$, we have the instance

$$\frac{(C, \mathtt{range}, B), (\mathtt{sp}, \mathtt{sp}, C), (A, \mathtt{sp}, A)}{(A, \mathtt{type}, B)}$$

– For the rule (6a) we have the instances $\dfrac{(A, \mathtt{sc}, A)}{(\mathtt{sc}, \mathtt{sp}, \mathtt{sc})}$ and $\dfrac{(A, \mathtt{sp}, A)}{(\mathtt{sp}, \mathtt{sp}, \mathtt{sp})}$.
Both instances have the same effect than the instances $\dfrac{}{(\mathtt{sc}, \mathtt{sp}, \mathtt{sc})}$ and $\dfrac{}{(\mathtt{sp}, \mathtt{sp}, \mathtt{sp})}$ of rule (6c)

The last assertion of the lemma follows directly form the above.

We will say that a rule r with instances R/R' *inserts* a name in ρdf if it appears in R' but not in R. Similarly, we will say that a rule r with instances R/R', different of a rule (1), *drops* a name p in ρdf if p appears in R but not in R'.

Next lemma needs no proof, because it is a simple matter of checking the ρdf vocabulary involved in each rule of Section 3.1.

Lemma 12. *Assume that all ρdf vocabulary mentioned in a triple appears as property.*
The only names in ρdf that are inserted (droped) by a rule are:

– $\mathtt{type}$, *inserted by rules* (4) *(droping $\mathtt{dom}$ or $\mathtt{range}$) and rules* (5) *(droping $\mathtt{dom}$, $\mathtt{range}$ and $\mathtt{sp}$).*
– $\mathtt{sc}$, *inserted in a triple* $(A, \mathtt{sc}, A)$ *by the rule* (7b) *(droping $\mathtt{dom}$, $\mathtt{range}$ or $\mathtt{type}$)*
– $\mathtt{sp}$, *inserted in a triple* $(A, \mathtt{sp}, A)$ *by the rules* (6a), (6c) *and* (6d) *(droping $\mathtt{dom}$ or $\mathtt{range}$), and droped by rule* (2b) *without the insertion of RDFS vocabulary.*

Let X be a fragment of ρdf and let G and H be X-graphs. We say that a rule R different of rules (1) is *superfluous* for X in a proof $G \vdash_{\rho\mathrm{df}} H$ if it inserts vocabulary outside X and the triples with the extra vocabulary inserted cannot be droped using rules different of (1) and must be deleted in some step only using the rule (1), in order to produce H. Hence none rule applied to the triples with extra RDFS vocabulary produce triples that influence H.

31

Lemma 13. *Assume that all ρdf vocabulary mentioned in a triple appears as property.*

The following are the unique cases of fragments X of ρdf where there are rules not included in $\mathrm{R}X$ which are superfluous for X:

1. $\mathrm{R}\{\mathtt{d}+\mathtt{r}\}$, *where rules* (4a) *and* (4b) *are superfluous and inserts* $\mathtt{type}$.
2. $\mathrm{R}\{\mathtt{d}+\mathtt{r}\}$ *and* $\mathrm{R}\{\mathtt{sp},\mathtt{d}+\mathtt{r}\}$, *where rules* (4a), (4b), (5a) *and* (5b) *are superfluous and insert* $\mathtt{type}$.
3. $\mathrm{R}\{\mathtt{type}\}, \mathrm{R}\{\mathtt{d}+\mathtt{r}\}, \mathrm{R}\{\mathtt{sp},\mathtt{type}\}, \mathrm{R}\{\mathtt{sp},\mathtt{d}+\mathtt{r}\}, \mathrm{R}\{\mathtt{type},\mathtt{d}+\mathtt{r}\}$, *where rule* (7b) *is superfluous and inserts* $\mathtt{sc}$.
4. $\mathrm{R}X$ *where* $\mathtt{sp} \notin X$, *where rules* (6a) *or* (6c) *or* (6d) *are superfluous and inserts* $\mathtt{sp}$.

Proof. Assume that X is a fragment of ρdf and that there exists a proof of $G \vdash_{\rho\mathrm{df}} H$, where G and H are two X-graphs, in which RDFS vocabulary outside X was inserted by a the application of some rule.

Lemma 12 now shows that the only possible cases are that where X does not contains $\mathtt{sc}$, $\mathtt{sp}$ or $\mathtt{type}$.

By lemma 12, the only rule which inserts $\mathtt{sc}$ is (7b), and adds a triple of the form $(A, \mathtt{sc}, A)$.

By Lemma 12, the only rules which insert $\mathtt{sp}$ are (6a), (6c) and (6d), and the triples added by the application of these rules as the form $(A, \mathtt{sp}, A)$.

By Lemma 12 the only rules that inserts $\mathtt{type}$ are (4) and (5).

But the only rule that drops $\mathtt{type}$ is (7b), which produces a triple of the form $(A, \mathtt{sc}, A)$.

We have two cases:

1. $\mathtt{type} \in X$. Hence $\mathtt{sc} \notin X$ or $\mathtt{sp} \notin X$ and we have a rule that inserts one of these names. So the triple $(A, \mathtt{sc}, A)$ or $(A, \mathtt{sp}, A)$ was added, but Lemma 11 and the assumption show that any rule other than (1) applied to $(A, \mathtt{sc}, A)$ or $(A, \mathtt{sp}, A)$ produces more triples of the same form. From this kind of triple, rule (1a) may add triples of the form $(X, \mathtt{sc}, Y)$ or $(X, \mathtt{sp}, Y)$, for $X \neq Y$ and at least one of them a blank; using such triples, we have that

   (a) rule (1a) produces the same kind of triples,
   (b) rules (2) for $\mathtt{sp}$ and (3a) for $\mathtt{sc}$ need to be applied or produces, the same kind of triples,
   (c) rules (6b) and (7) produce only triples of the form $(A, \mathtt{sc}, A)$ again,
   (d) rules (4) and (5) must needs to put a blank node as the predicate of a triple, which is not a well formed RDF triple.
   (e) rule (3b) applied to $(X, \mathtt{sc}, Y)$ has the form:

$$\frac{(X, \mathtt{sc}, Y), (Z, \mathtt{type}, X)}{(Z, \mathtt{type}, Y)}$$

   We have to analize two cases:

i. Y is blank. As rule (1a) has the instance

$$\frac{(Z, \texttt{type}, X)}{(Z, \texttt{type}, Y)}$$

we have that the rule that inserts $\texttt{sc}$ was innecesary in the proof.

ii. X is blank and Y is not a blank. We are assuming that X or Y cames from the application of rule (1a) to $(A, \texttt{sc}, A)$, so if Y is not a blank, we have $Y = A$. The mentioned instance has now the form

$$\frac{(X, \texttt{sc}, A), (Z, \texttt{type}, X)}{(Z, \texttt{type}, A)}$$

But from one or more applications of rule (1a) we have that $(Z, \texttt{type}, X)$ must be originated from a triple $(Z', \texttt{type}, A)$. If Z is a blank, it may be originated in one application of rule (1a); otherwise, $Z' = Z$. Hence under the application of rule (1a) to $(A, \texttt{sc}, A)$ we get the addition of the triples $(X, \texttt{sc}, A), (Z, \texttt{type}, X), (Z, \texttt{type}, A)$. Thus the rule inserting $\texttt{sc}$ is superfluous.

As $\texttt{sc} \notin \text{voc}(H)$ and $\texttt{sp} \notin \text{voc}(H)$, then the rules that inserts $\texttt{sc}$ or $\texttt{sp}$ are superfluous.

2. $\texttt{type} \notin X$. Hence X contains $\texttt{dom}$ or $\texttt{range}$ and may contains, for rule (5), $\texttt{sp}$. As rules (1) cannot be applied to get H and $\texttt{type} \notin \text{voc}(H)$, a rule that drops $\texttt{type}$ must exists. Lemma 12 shows that $\texttt{type}$ can only be droped by rule (7b), and as this rule adds a triple of the form $(A, \texttt{sc}, A)$, Lemma 11 shows that the only triples added thanks to the presence of $(A, \texttt{sc}, A)$ are triples of the form $(B, \texttt{sc}, B)$ or $(B, \texttt{sp}, B)$ or triples which mention $\texttt{type}$ again.

In both cases, we get that the rules used to insert RDFS vocabulary outside X are superflous.

Lemma 14. *Let X be a fragment ρdf and let G and H two X-graphs. Let $G = P_1, \ldots, P_k = H$ be a proof of $G \vdash_{\rho\text{df}} H$ If rules (1) are not used in this proof, then every rule used in the proof is in $\text{R}X$ or it is superfluous.*

Proof. We prove that any non X-rule is superflous or it cannot be applied in the proof. Hence only X-rules or superflous rules are applied.

Assume that a non X-rule r is used in some step in the proof, that is, there is j with $2 \leq j \leq k$ such that $P_{j-1} \vdash_r P_j$ and rule r has an instance $\frac{R}{R'}$ with R or R' mentioning RDFS vocabulary outside X. Suppose that p is the mentioned name, that is, $p \in \rho$df and $p \notin X$. If R mention p but P_{j-1} does not, then the rule is not applicable. So, if R mention RDFS vocabulary outside X, P_{j-1} does it, and if R' mention RDFS vocabulary outside X, P_j does it.

Moreover, note that none X-rule inserts p. Therefore, as G is an X-graph, it does not mention p, and inductively, assuming that every rule in the proof is an X-rule, we get that none of them inserts p, and we cannot have that p is mentioned anywhere in the proof.

Now assume first that p is mentioned in R and in P_{j-1}. By the above, p must be inserted in a previous step in the proof. Now we proceed to analize the rules that inserts RDFS vocabulary outside X. To do this, assume p is not mentioned in P_{j-1} and p is mentioned in P_j. By Lemma 12 we have two cases to analize:

1. $p = \texttt{type}$
   Hence $\texttt{dom}$ or $\texttt{range}$ belongs to X, and possibly $\texttt{sp} \in X$. But the unique rule other than (1) that drops $\texttt{type}$ is (7b) for $p = \texttt{type}$, which by Lemma (13) is superfluous for the proof.
2. $p = \texttt{sc}$ or $p = \texttt{sp}$
   By Lemma (13), the rules applied for this case are superfluous.

This completes the proof of the lemma.

Now we prove the theorem:

Lemma 14 shows that the only non superfluous rules are X-rules. Deleting from the proof all steps which are produced by non X-rules othar than rule (1), we obtain a proof using only X-rules and rule (1).

C.3   Proof of Lemma 1

Assume first that $G|_{\{\texttt{sp},A,B,C\}} \models_{\rho\mathrm{df}} (A,B,C)$. Clearly $G \models_{\rho\mathrm{df}} G|_{\{\texttt{sp},A,B,C\}}$ which gives $G \models_{\rho\mathrm{df}} (A,B,C)$.

To prove the converse, assume $G \models_{\rho\mathrm{df}} (A,B,C)$. Theorem 3 shows that $G \vdash_{\rho\mathrm{df}} (A,B,C)$. Thus there is a proof $P_1, \ldots, P_k$ of this. Analyzing the rules (2)-(7), the only rule which adds the triple (A,B,C) with $b \notin \rho\mathrm{df}$ is rule (2b). Hence the triples $(D,\texttt{sp},B)$ and (A,D,C) must be in P_{k-1}.

Note that using rule (6a) and rule (2b) we may have the following trivial situation:

$$\frac{(A,B,C)}{(B,\texttt{sp},B)}$$

$$\frac{(B,\texttt{sp},B),(A,B,C)}{(A,B,C)}$$

Delete from the proof any ocurrence of that situation.

Since (A,B,C) is ground, we have that, unless $(A,B,C) \in G$, the only rule applied to get this triple is (2b):

$$\frac{(B,\texttt{sp},D),(A,D,C)}{(A,B,C)}$$

With the deletion of trivial applications of rules, we have that $D \neq B$. Moreover, by assumption, D cannot be in $\rho\mathrm{df}$, because $B \notin \rho\mathrm{df}$ and $(B,\texttt{sp},D)$ is used.

Hence $(A,D,C) \in G$, or $(A,D,C) \notin G$ but $G \vdash_{\rho\mathrm{df}} (A,D,C)$ and we repeat the analisys. Note that (A,D,C) contains vocabulary from (A,B,C), and that any triple from where it may comes by rule (2b) also mention A and C.

On the other hand, $(B, \mathsf{sp}, D) \in G$, or $(B, \mathsf{sp}, D) \notin G$ and we have that only rule (2a) was applied

Note that using rule (2a) we may have the following trivial situations:

$$\frac{(B, \mathsf{sp}, B), (B, \mathsf{sp}, D)}{(B, \mathsf{sp}, D)}, \quad \frac{(B, \mathsf{sp}, D), (D, \mathsf{sp}, D)}{(B, \mathsf{sp}, D)}$$

Delete from the proof any ocurrence of both situations. Hence the application of rule (2a) is as:

$$\frac{(B, \mathsf{sp}, E), (E, \mathsf{sp}, D)}{(B, \mathsf{sp}, D)}$$

with $E \neq B$ and $E \neq D$.

As before, we get $E \notin \rho$df. Thus $(B, \mathsf{sp}, E) \in G$ or it is obtained from rule (2a), and the same holds for (E, sp, D). Note that (B, sp, E) and (E, sp, D) mention sp.

In any case, the only rules that may be applied are (2b) and (2a), and the triples from G used in the proof must mention A or B or C or sp. This proves the lemma.

C.4   Proof of Lemma 2

1. The implication from right to left is trivial. To prove the opposite direction, assume that $G \models (A, \mathsf{dom}, B)$. By Theorem 3, we have $G \vdash_{\rho \mathrm{df}} (A, \mathsf{dom}, B)$ and so there exists a proof $P_1, \ldots, P_k$ of this.
Now we concern with the rule of the step P_{k-1}, P_k. The only rule other than rule (1) that may produce (A, dom, B) is (2b) in the form

$$\frac{(Z, \mathsf{sp}, \mathsf{dom}), (A, Z, B)}{(A, \mathsf{dom}, B)}$$

   We have two cases:
   (a) $Z = \mathsf{dom}$, in which case $(A, \mathsf{dom}, B) \in P_{k-1}$, and so the application of (2b) was trivial, or
   (b) $Z \neq \mathsf{dom}$, in which case we contradicts the assumption of this secion about the use of ρdf volcabulary as subject or object of triples.
   Hence rule (2b) was not used in the step P_{k-1}, P_k, or it was trivial.
   Thus only rule (1) could be applied to P_{k-1} to produce P_k. But Lemma 10 implies that P_{k-1} must mention dom to produce (A, dom, B).
   Inductively, we get that G must mention dom to have proof of (A, dom, B).
   This proves the lemma.
2. The proof is similar.

C.5   Proof of Lemma 3

The implication from right to left is trivial. To prove the opposite direction, assume $A \neq B$ are ground terms and that $G \models (A, \mathsf{sc}, B)$. By Theorem 3, we have $G \vdash_{\rho \mathrm{df}} (A, \mathsf{sc}, B)$ and so there exists a proof $P_1, \ldots, P_k$ of this.

Now we concern with the rule of the step P_{k-1}, P_k. Rule (1a) cannot be used because the terms are ground, rule (2b) cannot be used because it imply the use of $\mathtt{sc}$ as subject or object in a triple, and it contradicts the general assumption of this section. Rules (2a), (3b), (4), (5) does not mention $\mathtt{sc}$ as predicate in their produced triples. Rules (6b) and (7) cannot be used because $A \neq B$. Thus only rules (1b) and (3a) produces $(A, \mathtt{sc}, B)$ with $A \neq B$ being ground terms. Rule (1b) means that $(A, \mathtt{sc}, B) \in P_{k-1}$, and we repeat the analisys now for P_{k-2} and P_{k-1}. Rule (3a) used has the form

$$\frac{(A, \mathtt{sc}, C), (C, \mathtt{sc}, B)}{(A, \mathtt{sc}, B)}$$

for some C. We have two cases:

1. C is ground. Hence $\{(A, \mathtt{sc}, C), (C, \mathtt{sc}, B)\} \subseteq P_{k-1}$ and we repeat the analisys now for P_{k-2} and P_{k-1}.
2. C is blank. We may have the following situation:

$$(7\mathrm{b}): \quad \frac{(Y, p, A)}{(A, \mathtt{sc}, A)}$$

with $p \in \{\mathtt{dom}, \mathtt{range}, \mathtt{type}\}$, and

$$(1\mathrm{a}): \quad \frac{(A, \mathtt{sc}, A)}{(A, \mathtt{sc}, C)}$$

and for $(C, \mathtt{sc}, B)$ we have

$$(7\mathrm{b}): \quad \frac{(Z, p, B)}{(B, \mathtt{sc}, B)}$$

with $p \in \{\mathtt{dom}, \mathtt{range}, \mathtt{type}\}$, and

$$(1\mathrm{a}): \quad \frac{(B, \mathtt{sc}, B)}{(C, \mathtt{sc}, B)}$$

In such cases we may use ρdf vocabulary different of $\mathtt{sc}$. But the use of rule (1a) uses the same blank, C, in different steps, to different terms, A and B. This fact is a violation of the allocation of blanks (see [13]), and also contradicts Theorem 4, because we cannot have a single application of rule (1a) with a map μ such that $\mu(C) = A \neq B = \mu(C)$.
Thus C is ground.

Thus the only triples in P_{k-1} which produce $(A, \mathtt{sc}, B)$ are of the form $(X, \mathtt{sc}, Y)$ with $X \neq Y$ being groud terms. Inductively, we have that the only triples in G that are used in the proof by a rule mention $\mathtt{sc}$. This proves the lemma.

The proof of $G \models (A, \mathtt{sp}, B) iff G|_{\mathtt{sp}} \models (A, \mathtt{sp}, B)$, for $A \neq B$ ground terms, is similar to that made for $\mathtt{sc}$, with $\mathtt{sc}$ replaced by $\mathtt{sp}$, and rule (3a) replaced by rule (2a)

C.6 Proof of Lemma 4

Assume $G \models H$. Theorem 3 shows that $G \vdash_{\rho\mathrm{df}} H$. Thus the re exists a proof $P_1, \ldots, P_k$ of this.

1. Suppose that $\mathtt{type} \notin \mathrm{voc}(H)$. By Theorem 4 any application of rule (1a) can be made in the last step. For P_{k-1}, as $\mathtt{type} \notin \mathrm{voc}(H)$, Lemmas 1, 2 and 3 shows that, to produce $\rho\mathrm{df}$ vocabulary in H we need the same vocabulary and, at most, we may force only to mention $\mathtt{sp}$ from G. Thus we have $G|_{\mathrm{voc}(H) \cup \{\mathtt{sp}\}} H$.
2. Suppose that $\mathtt{type} \in \mathrm{voc}(H)$ and that $\mathtt{sc} \notin \mathrm{voc}(H)$. The only differences with the previous proof are that
   (a) rules (4) and (5) may inserts $\mathtt{type}$ using $\rho\mathrm{df}$ vocabulary from the set $\{\mathtt{dom}, \mathtt{range}, \mathtt{sp}\}$.
   (b) As $\mathtt{sc} \notin \mathrm{voc}(H)$, we cannot apply rule (7b), which inserts $\mathtt{sc}$ from triples of the form (X, p, A) for $p \in \{\mathtt{dom}, \mathtt{range}, \mathtt{type}\}$. The problem with this rule is that it may uses some term X not included in $\mathrm{voc}(H) \cup \{\mathtt{dom}, \mathtt{range}, \mathtt{sp}\}$.
   In any case, we have that $G|_{\mathrm{voc}(H) \cup \{\mathtt{dom}, \mathtt{range}, \mathtt{sp}\}} \vdash_{\rho\mathrm{df}} H$, which proves the Lemma.

D Proofs of Section 5

D.1 Proof of Theorem 6

Proof. 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 $\rho\mathrm{df}$ the quadratic bound is trivial. The lower bound follows from the example below.

Example 2 (lower bound for the closure). Consider the graph $\{(a_1, \mathtt{sp}, a_2), \ldots, (a_n, \mathtt{sp}, a_{n+1})\} \cup \{(x_1, a_1, y_n), \ldots, (x_n, an, y_n)\}$. The number of triples of the closure of this graph is $2n + 1 + \sum_{k=1}^{n} k$, that is order $\Omega(n^2)$.

D.2 Proof of Theorem 7

Proof (Sketch). 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 1 and rule 2(a). The construction of $G(\mathtt{sp})^*$ can be done in $|G| \log |G|$ steps: order $G_\emptyset$ and then while traversing $G(\mathtt{sp})$ do binary search on $G_\emptyset$. For point 6 (see Figure 3) the crucial observation is that in $G(\mathtt{sp})'$, 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 $G \models (a, \mathtt{type}, v)$ 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 $G(\mathtt{sc})'$ will do the job.
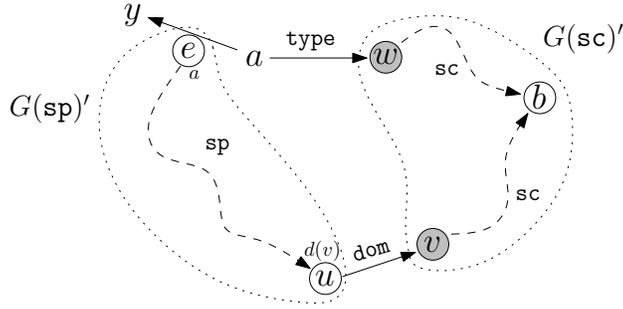
Figure 3. Point 6 of the *Ground Entailment Algorithm*

D.3 Proof of Corollary 3

Proof. Just note that for ground graph H, $G \models_{\rho df} H$ iff for each $t \in H$, $G \models_{\rho df} t$.

D.4 Proof of Proposition 2

Proof. The bound is obtained by coding the problem of determining if given sets A, B, $A \cap B \neq \emptyset$. Given $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$, construct an RDF graph as follows: $G = \{(a_{i-1}, \mathtt{sp}, a_i)\}_{2 \leq i \leq n} \cup \{(x, b_j, y)\}_{1 \leq j \leq n}$. Then use the fact that $G \models (x, a_n, y)$ iff $A \cap B \neq \emptyset$.