Simple and Efficient Minimal RDFS [1]

Sergio Munoz-Venegas [a], Jorge Pérez [b,d], Claudio Gutierrez [c,d]

[a] *Department of Mathematics, Universidad Chile*
[b] *Department of Computer Science, Pontificia Universidad Católica de Chile*
[c] *Department of Computer Science, Universidad Chile*
[d] *Khipu: South Andean Center for Database Research*

Abstract

The original RDFS language design includes several features that hinder the task of developers and theoreticians. This paper has two main contributions in the direction of simplifying the language. First, it introduces a small fragment which, preserving the normative semantics and the core functionalities, avoids the complexities of the original specification, and captures the main semantic functionalities of RDFS. Second, it introduces a minimalist deduction system over this fragment, which by avoiding certain rare cases, obtains a simple deductive system and a computationally efficient entailment checking.

Key words: RDF, RDFS, Efficient entailment, Semantic Web

1. Introduction

The Resource Description Framework (RDF) is the W3C standard for representing information in the Web [12]. The motivation behind the development of RDF by the W3C was to have a common and minimal language to enable to map large quantities of existing data onto it so that the data can be analyzed in ways never dreamed of by its creators [3]. Bringing this vision to reality amounts to make the processing of RDF data at big scale viable.

Efficient processing of any kind of data relies on a compromise between 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 current developments show (e.g. DBpedia [8], FOAF [7], Gene Ontology [13], etc. For studies on data sets see [19]).

Hence, a program to make RDF processing scalable has to consider necessarily the compromise between complexity and expressiveness. Such a program amounts essentially to look for fragments of RDF with good behavior with respect to complexity of processing. This is the broad goal of the present paper.

The RDF specification is given in [11] and its semantics is defined in [17] and has been studied from several points of view [16,9,20,24]. Essentially, an RDF statement is a subject-predicate-object structure, called an RDF *triple*, intended to describe resources and properties of those resources. Subject and object of an RDF triple can be anonymous resources, known as *blank nodes*. An RDF graph (or simply a graph) is a set of RDF triples. In addition to this basic structure, the RDF specification includes a built-in vocabulary, the RDFS vocabulary, that deal with inheritance of classes and properties, as well as typing, among other features [11] [2].

Email addresses: smunozv@uchile.cl (Sergio Munoz-Venegas), jperez@ing.puc.cl (Jorge Pérez), cgutierr@dcc.uchile.cl (Claudio Gutierrez).

[1] This is an extended and revised version of [21].

[2] In this paper we call RDFS vocabulary to the set all URIs under the prefixes (*namespaces*) *rdf:* and *rdfs:* in [17].

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 normative RDF Semantics specification were not complete [20,24]. A second empirical observation is that several parts of the RDFS vocabulary have been deprecated, 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. These and other concerns have led researchers to formulate different proposals of interesting fragments or closely related specifications [25,22,15,4] improving diverse aspects of the language (we will discuss them in the section of related work). Another example where this type of issues arises is the SPARQL query language specification [23], that currently does not support RDFS entailment. 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, and to know exactly what are the minimal portions of the whole vocabulary that interplay in the entailment of every specific fragment. In this paper we address these issues.

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

(i) To identify a fragment which encompasses the essential features of RDFS, which preserves the original semantics, be easy to formalize and can serve to prove results about its properties.

(ii) To point out features that the groups involved in the development and standardization of RDFS and its query language should take a more careful look at, and suggest directions for improvements.

(iii) To study the complexity of entailment for the vocabulary in general and in these fragments in particular, and to develop efficient algorithms for testing entailment.

As for the first point, in this paper we isolate a fragment of RDFS that covers the most relevant vocabulary, prove that it preserves the original RDFS 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. It is composed of the reserved vocabulary *rdfs:subClassOf, rdfs:subPropertyOf, rdf:type, rdfs:domain* and *rdfs:range*. The rest of the RDFS vocabulary has either light or no semantics at all, and much of it plays the role of structural information about the internals of the language itself. A good example of this latter function is the keyword *rdfs:Resource*. An exception is *rdfs:Class*, which is important for the ontological level of the Semantic Web, but, as we show in this paper, in the frame of RDFS deductions does not play any relevant role. We lift this structural information into the semantics of the language, hiding it from developers and users, and present a complete and sound deduction system for the fragment.

Furthermore, we study the subfragments of the core fragment, prove that they retain the original RDFS semantics, and present results that identify the minimal parts of the vocabulary needed in the deductions of every subfragment.

Regarding the second point, we study possible refinements of the fragment presented in the first item. In particular, we show that reflexivity of *rdfs:subClassOf* and *rdfs:subPropertyOf* included in the official RDF specification [17], play no relevant role in the semantics and thus can be avoided without having side-effects. This greatly simplifies the deductive system. A more delicate issue we address is the occurrence of RDFS vocabulary in subject and object positions. These occurrences rarely occur in practice and from a theoretical point of view are re-definitions of the original semantics.

Considering the above arguments (and some complexity issues discussed below) we introduce a fragment, which we call *minimal RDFS*, based on the restricted vocabulary mentioned above, avoiding reflexivity of *rdfs:subClassOf* and *rdfs:subPropertyOf*, not having distinguished vocabulary in subject nor object positions, and including only *ground triples*, that is, triples without blank nodes. We show that this fragment behaves semantically well; in fact, we give a semantics for it, and present a simple deductive system sound and complete. Our goal is that this fragment should work as an essential minimal and efficient floor from where to extend the language in different directions. For example, in this paper we show a general result that states that deductions can be *normalized* into a ground and a non-ground part. Thus, deductions that need blank nodes semantics, can be modularly composed with deductions for our minimal fragment. This result is also the theoretical basis of why we consider only ground triples in the minimal RDFS fragment.

Concerning the third item, complexity of entailment, the minimal RDFS system has several good properties. For testing RDFS entailment in the ground case even current known bounds seems totally impractical. For example, the naive approach would use *closure* of graphs. The closure is a completion of the graph obtained by adding all the triples that are entailed by the graph. Estimates for the size of the closure are high: we show that the size of the closure is quadratic in the worst case. This bound is impractical from a database point of view. In this paper we present an algorithm for entailment which shows that, testing whether a ground triple is entailed by a graph G, can be decided in time $\mathcal{O}(|G| \log |G|)$ in the worst case, where $|G|$ stands for the number of triples of G. We also prove that this is a tight bound for the complexity of ground entailment.

Finally, we extend the previous result to a large family of non-ground graphs of practical significance. In fact, we prove that for graphs G and H with H containing at most one blank node per triple, testing whether H is entailed by G can be done in time $\mathcal{O}(|H||G| \log |G|)$.

The rest of the paper is organized as follows. Section 2 presents preliminary material. In Section 3 we introduce the fragment ρdf of RDFS, which is the base for further developments, and prove that it is self-contained and preserves the original semantics of RDFS. We present a notion of proof which is sound and complete for the semantics. Section 4 introduces *minimal RDFS*, a minimal fragment based on a small vocabulary, which avoids reflexivity constraints, RDFS vocabulary in subject and object positions, and consider only ground triples. We present arguments showing that this system behaves computationally well and has very simple entailment rules. We present an algorithm which performs entailment efficiently on this fragment. In order to easy the reading of the paper, proofs of results in Section 3 and 4 which are long and not necessary to follow the main arguments of the paper are placed in the appendix. For the sake of completeness, the appendix also includes parts of the official RDF specification that are relevant to this work.

2. Preliminaries

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 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 [5,23]– we are omitting the usual 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 set of* terms *of a graph G, denoted by* terms(G) *is the set of elements of* **UBL** *that occur in the triples of G. The* vocabulary *of G, denoted by* voc(G) *is the set* terms(G) $\cap$ **UL**. *A graph is* ground *if it has no blank nodes.*

A *vocabulary* is a subset of **UL**. Given a vocabulary V and an RDF graph G, we say that G is a graph *over* V whenever voc(G) $\subseteq V$.

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

The RDF specification [17] includes a set of reserved names, the RDFS vocabulary (RDF Schema [11]) designed to describe relationships between resources as well as to describe properties like attributes of resources (traditional attribute-value pairs). Table A.1 (in the appendix) shows the full RDFS vocabulary [17], and (in brackets) the shortcuts that we use in this paper. We assume that the set **U** includes the RDFS vocabulary. Also notice that we call RDFS vocabulary to all the URIs under the prefixes (namespaces) *rdf:* and *rdfs:* in [17] (that is, we assume in this paper that the RDFS vocabulary already contains the RDF vocabulary [17]). Notice that our definition of the RDFS vocabulary also contains the keyword *rdf:XMLLiteral*, and thus, we consider XML typed literals part of the RDFS vocabulary.

2.1. *Interpretations and RDFS semantics*

The normative semantics for RDF graphs given in [17] follows standard classical treatment in logic defining the notions of model, interpretation, entailment, and so on. The RDFS theory is built incrementally from *Simple interpretations*, to *RDF interpretations*, and to *RDFS interpretations* [17]. In

this paper we use a single notion of interpretation which summarizes Simple, RDF, and RDFS interpretations in one step. In order to concentrate on the core semantics, we do not include *Datatyped interpretations* in this work.

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) $Ext : Prop \rightarrow 2^{Res \times Res}$, a mapping that assigns an* extension *to each property name; (5) $CExt : Class \rightarrow 2^{Res}$ a mapping that assigns a set of resources to every resource denoting a class; (6) $Lit \subseteq Res$ the set of literal values that contains all the plain literals in $\mathbf{L} \cap V$; (7) $Int : \mathbf{UL} \cap V \rightarrow Res \cup Prop$, 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 [17] the semantics of RDF graphs is defined by using the notion of *entailment* based on the idea of *satisfaction* of a graph under a given interpretation. Intuitively, given a vocabulary V and an interpretation $\mathcal{I}$ over V, a ground triple (s, p, o) over V is *true* under $\mathcal{I}$ if:

- $\mathcal{I}$ *interprets* p as a property name, and thus, $\mathcal{I}$ assigns an *extension* (a set of pairs of resources) to the interpretation of p, and
- the interpretation of the pair (s, o) belongs to the extension of the interpretation of p.

Notice that, since the set of resources and the set of property names of an interpretation are not necessarily disjoint, an element in V can be simultaneously interpreted as a resource and as a property name. This feature reflects the capability of RDF of stating a predicate about properties, that is, properties may occur as subjects or objects in triples.

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 an element 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, an extension of the interpretation map Int is used. Let $A : \mathbf{B} \rightarrow Res$ be a function from blank nodes to resources. We denote by Int_A the extension of Int that includes blanks as part of its domain and is 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 graphs that includes RDFS vocabulary 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 formal definition of the normative semantics of RDFS using the notion of interpretation given in Definition 2.

3. The ρdf fragment of RDFS

In this section we define the fragment of RDFS that we study in this paper. We define a semantics and a set of inference rules for this fragment, proving completeness and soundness, and proving that these rules captures the RDFS semantics for the fragment.

The fragment to be considered comprises the RDFS keywords (with shortcuts in brackets) *rdfs:subPropertyOf* [`sp`], *rdfs:subClassOf* [`sc`], *rdfs:domain* [`dom`], *rdfs:range* [`range`] and *rdf:type* [`type`]. This fragment is relevant for several reasons. The intended meaning of the fragment is non-trivial and is designed to relate individual pieces of data external to the vocabulary of the language, thus having a deep semantical role in RDFS. Their semantics can be defined by rules which involve variables (to be instantiated by real data). For example, `sc` is a binary property reflexive and transitive; when combined with `type`, it specifies that the type of an individual (a class) can be lifted to that of a superclass. Moreover, as we show in this section, the above fragment is *self-contained*: the entailment relation between RDFS graphs that only mention vocabulary of this fragment does not rely on vocabulary outside the fragment (see Theorem 5). As our results show, there are theoretical reasons that support the convenience of the choice of the fragment.

On the other hand, the predicates left out have a light semantics essentially describing its internal function in the ontological design of the system of classes of RDFS. Most of their semantics is defined by *axiomatic triples* [17] (see Table A.2 in the appendix), 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 se-

mantics 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.

Definition 3 *Define* ρdf *vocabulary*[3] *to be the following subset of the RDFS vocabulary:*

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

An RDFS triple is called a ρdf*–triple if all the RDFS vocabulary that is mentioned in the triple belongs to* ρdf*. A* ρdf*–graph is defined as a set of* ρdf*–triples.*

3.1. *Semantics for* ρdf

Definition 4 *Let G be a* ρdf*–graph. An interpretation $\mathcal{I} = (Res, Prop, Class, Ext, CExt, Lit, Int)$ is a* model *of G under* ρdf *(*ρdf*–model for short), denoted $\mathcal{I} \models_{\rho\text{df}} G$, iff $\mathcal{I}$ is an interpretation over* ρdf∪terms(G) *that satisfies the following conditions:*

(i) Simple:
- *(a) there exists a function $A : \mathbf{B} \to Res$ such that for each $(s, p, o) \in G$, it holds that $Int(p) \in Prop$ and $(Int_A(s), Int_A(o)) \in Ext(Int(p))$, where Int_A is the extension of Int using A.*

(ii) Subproperty:
- *(a) $Ext(Int(\texttt{sp}))$ is transitive and reflexive over $Prop$*
- *(b) if $(x, y) \in Ext(Int(\texttt{sp}))$ then $x, y \in Prop$ and $Ext(x) \subseteq Ext(y)$*

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

(iv) Typing I:
- *(a) $x \in CExt(y)$ iff $(x, y) \in Ext(Int(\texttt{type}))$*
- *(b) if $(x, y) \in Ext(Int(\texttt{dom}))$ and $(u, v) \in Ext(x)$ then $u \in CExt(y)$*
- *(c) if $(x, y) \in Ext(Int(\texttt{range}))$ and $(u, v) \in Ext(x)$ then $v \in CExt(y)$*

(v) Typing II:
- *(a) For each $\texttt{e} \in$ ρdf, $Int(\texttt{e}) \in Prop$.*
- *(b) if $(x, y) \in Ext(Int(\texttt{dom}))$ then $x \in Prop$ and $y \in Class$.*
- *(c) if $(x, y) \in Ext(Int(\texttt{range}))$ then $x \in Prop$ and $y \in Class$.*
- *(d) if $(x, y) \in Ext(Int(\texttt{type}))$ then $y \in Class$.*

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

[3] Read rho-df, the ρ from *restricted* rdf.

Notice that in ρdf–models we do not impose the *a priori* satisfaction of any axiomatic triple. Indeed, ρdf–models do not satisfy any of the RDFS axiomatic triples given in [17] because all of them mention RDFS vocabulary outside ρdf. This is also the reason for the inclusion of conditions (4) in ρdf models because they capture semantically the restrictions imposed syntactically by the RDFS axiomatic triples (dom, dom, prop), (dom, range, class), (range, dom, prop), (range, range, class), and (type, range, class), and the fact that every element in ρdf must be interpreted as a property.

The next theorem shows that this definition retains the normative RDFS semantics for the ρdf vocabulary.

Theorem 5 *Let $\models$ be the RDFS entailment defined in [17], and let G and H be* ρdf*–graphs. Then*

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

PROOF. We present a brief sketch of the proof. The complete proof can be found in Appendix B[4]. In [17], the RDFS entailment relation is defined by using the notion of *RDFS-model*: given G and H, it holds that $G \models H$ if every RDFS-model of G is also an RDFS-model of H (see Definition 30 in the appendix). An RDFS-model must satisfy several *semantics conditions* imposed to properly interpret the RDFS vocabulary (see Definition 30 in the appendix). It is straightforward to show that if $\mathcal{I}$ is an RDFS-model of a ρdf graph G, then $\mathcal{I}$ is also a ρdf-model of G. The crucial property is that if $\mathcal{I}$ is a ρdf-model of a ρdf-graph G and $\mathcal{I}$ satisfies all the semantic conditions interpret the RDFS vocabulary, then $\mathcal{I}$ is also an RDFS-model of G. The "if" part of the theorem follows from this last fact. Assume that $G \models_{\rho\text{df}} H$ and that $\mathcal{I}$ is an RDFS-model of G. We need to prove that $\mathcal{I}$ is an RDFS-model of H. Since $\mathcal{I}$ is an RDFS-model of G, we know that $\mathcal{I}$ is also a ρdf-model of G and, therefore, from $G \models_{\rho\text{df}} H$ we obtain that $\mathcal{I}$ is a ρdf-model of H. Finally, $\mathcal{I}$ is a ρdf-model of H that satisfies the semantic conditions imposed to RDFS-models, and thus, $\mathcal{I}$ is an RDFS-model of H.

The proof of the "only if" part is more involved and relies on the fact that given an interpretation $\mathcal{I}$ one can build a new interpretation $\mathcal{I}'$ that satisfies

[4] To give continuity of reading to the main results of the paper, we send proofs of theorems (when long and technical) to Appendix B.

the following property: given a ρdf-graph G, the interpretation $\mathcal{I}$ is a ρdf-model of G if and only if $\mathcal{I}'$ is an RDFS-model of G. Thus, assume that $G \models H$ and let $\mathcal{I}$ be a ρdf-model of G. We need to prove that $\mathcal{I}$ is a ρdf-model of H. Since $\mathcal{I}$ is a ρdf-model of G, we know that $\mathcal{I}'$ is an RDFS-model of G, and thus, from $G \models H$ we obtain that $\mathcal{I}'$ is an RDFS-model of H. Finally by the properties of $\mathcal{I}$ and $\mathcal{I}'$, we know that $\mathcal{I}$ is a ρdf-model of H which was to be proved. □

3.2. *A deductive system for the* ρdf *fragment*

Now we present a deductive system for ρdf based on a system of rules for RDFS entailment defined in the W3C Recommendation [17]. Loosely speaking, we work with those rules in [17] which only involves ρdf vocabulary, and that do not involve allocation of blanks to literals nor datatypes. We prove later that the deductive system defined for the ρdf fragment is sound and complete for ρdf–entailment. Table 1 shows the rules for the fragment ρdf.

It is worth mentioning that, as noted in [20,24], the set of rules presented in [17] is not complete for RDFS entailment. The problem with the system proposed in [17] is that a blank node X can be implicitly used as a property in triples like $(a, \mathtt{sp}, X)$, $(X, \mathtt{dom}, b)$, and $(X, \mathtt{range}, c)$. In this paper we follow the solution proposed by Marin [20]. In fact, the rules (5a)-(5b) were added in [20] to the system given in [17] to deal with this problem.

Now we formalize deductions made with rules (1)-(7). In every rule letters $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}$, and $\mathcal{Y}$, stand for variables to be replaced by actual terms. More formally, an *instantiation* of a rule is a uniform replacement of the variables 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. For example, if $a, b \in \mathbf{U}$, $N \in \mathbf{B}$ and $y \in \mathbf{L}$, then $\frac{R}{R'}$ with $R = \{(a, \mathtt{sp}, b), (N, a, y)\}$ and $R' = \{(N, b, y)\}$, is an instantiation of rule (2b).

Definition 6 (Proof) *Let G and H be RDFS 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 $P_j = P_{j-1} \cup R'$.*

Such sequence of graphs is called a proof *of $G \vdash_{\rho\mathrm{df}} H$. Whenever $G \vdash_{\rho\mathrm{df}} H$, we say that the graph H is* derived *from the graph G. Each pair (P_{j-1}, P_j), $1 \leq j \leq k$ is called a* step *of the proof which is labeled by the respective instantiation $\frac{R}{R'}$ of the rule applied at the step.*

The relation $\vdash_{\rho\mathrm{df}}$ is well defined for ρdf–graphs in the sense that from a ρdf–graph G only ρdf–graphs can be proved by using rules (1)–(7):

Proposition 7 *Let G be a ρdf–graph and let H be an RDFS graph. Assume $G \vdash_{\rho\mathrm{df}} H$. Then H is a ρdf–graph.*

PROOF. Suppose the assertion of the proposition is false. Thus there are some triples in H with RDFS vocabulary not included in ρdf occurring as subject or object. Let $P_1, P_2, \ldots, P_k$ be a proof of $G \vdash_{\rho\mathrm{df}} H$, with $P_1 = G$ and $P_k = H$. Since G is a ρdf–graph, it follows recursively that exists j with $1 < j \leq k$ such that P_{j-1} is a ρdf–graph and there is some RDFS vocabulary occurring in P_j that does not belong to ρdf.

As the application of rule (1) at most adds to P_{j-1}, if adds any, only triples with some URIs replaced by blank nodes, we have that the occurrence of RDFS vocabulary either does not change, or decrease. Therefore rule (1) was not applied to the step (P_{j-1}, P_j) of the proof. Hence there is an instance $\frac{R}{R'}$ of a rule (2)-(7) that was applied to the step (P_{j-1}, P_j) of the proof. Thus $P_j = P_{j-1} \cup R'$ with $R \subseteq P_{j-1}$ and with some RDFS vocabulary out of ρdf occurring in triples of R'. But a simple inspection of these rules shows that R must have also some RDFS vocabulary out of ρdf occurring in its triples. As $R \subseteq P_{j-1}$, we have that P_{j-1} is not a ρdf graph, a contradiction. □

Theorem 8 (Soundness and completeness) *Let G and H be ρdf graphs. Then*

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

The proof of the above theorem is a long list of checkings and it is given in Appendix B. From Theorem 5 and Theorem 8 we get the following corollary.

Corollary 9 *Let $\models$ be the RDFS entailment defined in [17], and let G and H be ρdf graphs. Then*

$$G \models H \;\; \textit{iff} \;\; G \vdash_{\rho\mathrm{df}} H.$$

Thus, the set of rules (1)–(7) captures RDFS entailment when restricted to ρdf vocabulary.

The following result follows from the proof of Theorem 8. Its proof is given in Appendix B.

(1) Simple:		
	(a) $\frac{G}{G'}$ for a map $\mu : G' \to G$	(b) $\frac{G}{G'}$ for $G' \subseteq G$
(2) **Subproperty**:		
	(a) $\frac{(\mathcal{A},\texttt{sp},\mathcal{B})\ \ (\mathcal{B},\texttt{sp},\mathcal{C})}{(\mathcal{A},\texttt{sp},\mathcal{C})}$	(b) $\frac{(\mathcal{A},\texttt{sp},\mathcal{B})\ \ (\mathcal{X},\mathcal{A},\mathcal{Y})}{(\mathcal{X},\mathcal{B},\mathcal{Y})}$
(3) **Subclass**:		
	(a) $\frac{(\mathcal{A},\texttt{sc},\mathcal{B})\ \ (\mathcal{B},\texttt{sc},\mathcal{C})}{(\mathcal{A},\texttt{sc},\mathcal{C})}$	(b) $\frac{(\mathcal{A},\texttt{sc},\mathcal{B})\ \ (\mathcal{X},\texttt{type},\mathcal{A})}{(\mathcal{X},\texttt{type},\mathcal{B})}$
(4) **Typing**:		
	(a) $\frac{(\mathcal{A},\texttt{dom},\mathcal{B})\ \ (\mathcal{X},\mathcal{A},\mathcal{Y})}{(\mathcal{X},\texttt{type},\mathcal{B})}$	(b) $\frac{(\mathcal{A},\texttt{range},\mathcal{B})\ \ (\mathcal{X},\mathcal{A},\mathcal{Y})}{(\mathcal{Y},\texttt{type},\mathcal{B})}$
(5) Implicit Typing:		
	(a) $\frac{(\mathcal{A},\texttt{dom},\mathcal{B})\ \ (\mathcal{C},\texttt{sp},\mathcal{A})\ \ (\mathcal{X},\mathcal{C},\mathcal{Y})}{(\mathcal{X},\texttt{type},\mathcal{B})}$	(b) $\frac{(\mathcal{A},\texttt{range},\mathcal{B})\ \ (\mathcal{C},\texttt{sp},\mathcal{A})\ \ (\mathcal{X},\mathcal{C},\mathcal{Y})}{(\mathcal{Y},\texttt{type},\mathcal{B})}$
(6) Subproperty Reflexivity:		
	(a) $\frac{(\mathcal{X},\mathcal{A},\mathcal{Y})}{(\mathcal{A},\texttt{sp},\mathcal{A})}$	(c) $\frac{}{(p,\texttt{sp},p)}$ for $p \in \rho\text{df}$
	(b) $\frac{(\mathcal{A},\texttt{sp},\mathcal{B})}{(\mathcal{A},\texttt{sp},\mathcal{A})\ \ (\mathcal{B},\texttt{sp},\mathcal{B})}$	(d) $\frac{(\mathcal{A},p,\mathcal{X})}{(\mathcal{A},\texttt{sp},\mathcal{A})}$ for $p \in \{\texttt{dom},\texttt{range}\}$
(7) Subclass Reflexivity:		
	(a) $\frac{(\mathcal{A},\texttt{sc},\mathcal{B})}{(\mathcal{A},\texttt{sc},\mathcal{A})\ \ (\mathcal{B},\texttt{sc},\mathcal{B})}$	(b) $\frac{(\mathcal{X},p,\mathcal{A})}{(\mathcal{A},\texttt{sc},\mathcal{A})}$ for $p \in \{\texttt{dom},\texttt{range},\texttt{type}\}$

Table 1
Deductive rules for the fragment ρdf.

Theorem 10 (Normal form for proofs) *Let G and H be ρdf–graphs and assume that $G \vdash_{\rho\text{df}} H$. Then there is a proof of H from G (in the sense of Definition 6) such that rule (1a) is used at most once and at the last step of the proof.*

Notice that rule (1a) captures the semantics of blank nodes. Thus, Theorem 10 essentially states that the inference regarding blank nodes can always be postponed to the last step in a proof of entailment. We use this result in Section 4 when proposing a minimalist inference system for RDFS graphs.

3.3. *The role of reflexivity*

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 $(\texttt{sp},\texttt{sp},\texttt{sp})$, $(\texttt{sc},\texttt{sp},\texttt{sc})$, $(\texttt{type},\texttt{sp},\texttt{type})$, $(\texttt{dom},\texttt{sp},\texttt{dom})$, and $(\texttt{range},\texttt{sp},\texttt{range})$. These triples are true under every ρdf model since sp must be interpreted as a reflexive relation. Moreover, since blank nodes work as existential variables, the triples above with their subject or object positions replaced by blank nodes, 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 as the following proposition shows.

Proposition 11 *Let t be an RDF triple such that $\emptyset \models_{\rho\text{df}} t$. Then, either $t \in \{(\texttt{sp},\texttt{sp},\texttt{sp}), (\texttt{sc},\texttt{sp},\texttt{sc}), (\texttt{type},\texttt{sp},\texttt{type}), (\texttt{dom},\texttt{sp},\texttt{dom}), (\texttt{range},\texttt{sp},\texttt{range})\}$, or t is obtained from these triples replacing the subject or/and the object by blank nodes.*

PROOF. It is not difficult to see that the triples in the set $A = \{(\texttt{sp},\texttt{sp},\texttt{sp}), (\texttt{sc},\texttt{sp},\texttt{sc}), (\texttt{type},\texttt{sp},\texttt{type}), (\texttt{dom},\texttt{sp},\texttt{dom}), (\texttt{range},\texttt{sp},\texttt{range})\}$ together with their existential versions obtained by replacing subject or predicate by blank nodes, are satisfied by every ρdf model, and so they are ρdf–entailed by every graph G. The rest of the proof follows by a case by case analysis, taking into account that, for a ground triple to be satisfied by

every model, all its components must be elements in ρdf. $\square$

We show next that this is part of a more general phenomena, namely the presence of reflexivity for sp and sc. In fact, we prove that reflexivity for sp and sc is orthogonal to the rest of the semantics.

Definition 12 *An interpretation $\mathcal{I}$ is a* reflexive–relaxed *model under ρ*df *of a graph G, denoted by $\mathcal{I} \models^{\text{nrx}}_{\rho\text{df}} G$, iff $\mathcal{I}$ is an interpretation that satisfies the conditions in Definition 4 for G with the exception that $\mathcal{I}$ does not necessarily satisfy the restrictions stating that $Ext(Int(\texttt{sp}))$ and $Ext(Int(\texttt{sc}))$ are reflexive relations over Prop and Class, respectively.*

For graphs G and H we write $G \models^{\text{nrx}}_{\rho\text{df}} H$, iff every reflexive-relaxed model of G is also a reflexive-relaxed model of H.

Theorem 13 *Let G and H be ρdf graphs. Assume that H does not contain triples of the form $(x, \texttt{sp}, x)$ nor $(x, \texttt{sc}, x)$ for $x \in$ **UL**, nor triples obtained from them by replacing the subject and/or the object by blank nodes. Then*

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

PROOF. $\Leftarrow$) Assume that $G \models^{\text{nrx}}_{\rho\text{df}} H$ and let $\mathcal{I} = (Res, Prop, Class, Ext, CExt, Lit, Int)$ be a ρdf model of G. We need to show that $\mathcal{I}$ is a ρdf model of H. Since $\mathcal{I}$ is a ρdf model of G, by definition we know that $\mathcal{I}$ is also a reflexive–relaxed ρdf model of G. Now, since $G \models^{\text{nrx}}_{\rho\text{df}} H$, we have that $\mathcal{I}$ is also a reflexive–relaxed ρdf model for H. Now, $\mathcal{I}$ is an interpretation that satisfies the conditions of Definition 12 for H, and is such that $Ext(Int(\texttt{sp}))$ and $Ext(Int(\texttt{sc}))$ are reflexive relations. Hence, $\mathcal{I}$ satisfies all the conditions of Definition 4 for H and so $\mathcal{I}$ is a model under ρdf for H, completing this part of the proof.

$\Rightarrow$) Assume that $G \models_{\rho\text{df}} H$ and let $\mathcal{I} = (Res, Prop, Class, Ext, CExt, Lit, Int)$ be a reflexive-relaxed ρdf model of G. We need to show that $\mathcal{I}$ is a reflexive-relaxed ρdf model of H. Let $\mathcal{I}'$ be the model obtained from $\mathcal{I}$ by taking the reflexive closure of the relations $Ext(Int(\texttt{sp}))$ over $Prop$, and $Ext(Int(\texttt{sc}))$ over $Class$. Thus $\mathcal{I}'$ is a ρdf model of G (it satisfies all the conditions in Definition 4 for G). Since $G \models_{\rho\text{df}} H$, we have that $\mathcal{I}'$ is a ρdf model for H. Now we 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. Thus $(Int_A(s), Int_A(p)) \in Ext(Int(p))$ for every $(s, p, o) \in H$. We also know that $\mathcal{I}$ and $\mathcal{I}'$ differ only in the reflexive pairs of $Ext(Int(\texttt{sp}))$ and $Ext(Int(\texttt{sc}))$. Now, because H does not contain triples of the form $(x, \texttt{sp}, x)$ nor $(x, \texttt{sc}, x)$ nor their existential versions replacing subject or object by blank nodes, the same extension function A is such that in $\mathcal{I}$ it holds that $(Int_A(s), Int_A(p)) \in Ext(Int(p))$ for every $(s, p, o) \in H$, hence it satisfies all the conditions of Definition 12, and finally $\mathcal{I}$ is a reflexive–relaxed model for H, completing this part of the proof. $\square$

Essentially the above theorem states that the only use of reflexive restrictions in RDFS models is the entailment of triples of the form $(x, \texttt{sp}, x)$, $(x, \texttt{sc}, x)$ (or their existential versions).

Another property of $\models^{\text{nrx}}_{\rho\text{df}}$ is that it does not entail axiomatic triples:

Corollary 14 *There is no triple t such that*

$$\emptyset \models^{\text{nrx}}_{\rho\text{df}} t.$$

PROOF. It is evident that, as the interpretation of sp is not necessarily reflexive over property names, we see that none of the triples in $A = \{(\texttt{sp}, \texttt{sp}, \texttt{sp}), (\texttt{sc}, \texttt{sp}, \texttt{sc}), (\texttt{type}, \texttt{sp}, \texttt{type}), (\texttt{dom}, \texttt{sp}, \texttt{dom}), (\texttt{range}, \texttt{sp}, \texttt{range})\}$ are axiomatic for $\models^{\text{nrx}}_{\rho\text{df}}$. Finally, the fact that $\models^{\text{nrx}}_{\rho\text{df}} \subseteq \models_{\rho\text{df}}$ completes the proof. $\square$

4. Minimal RDFS

Our main goal in this work is to study the core process of entailment in RDFS. Towards our goal we have presented the ρdf fragment, and we have formally proved that the normative RDFS entailment restricted to this fragment is captured by the inference rules (1)–(7). In this section we take a step forward in our investigation defining the fragment of *minimal RDFS*. We present a system of rules for this fragment proving that, avoiding reflexivity, the system is sound and complete for RDFS entailment for minimal RDFS graphs. We also prove tight complexity bounds for the problem of testing entailment, and provide an algorithm that is asymptotically optimal to test entailment in the fragment.

The refinement of ρdf involves restricting graphs to be ground, and rejecting the presence of ρdf vocabulary as subject or object of the triples. Avoiding the presence of ρdf vocabulary as subject or object of triples implies that we avoid the redefinition of RDFS vocabulary. In [10] the presence of RDFS vocabulary as subject or object of triples was called

"non-standard use of RDFS vocabulary". We agree with [10] conjecturing that, in practice, large classes of RDFS graphs will not have RDFS vocabulary occurring as subject or object in their triples. On the other hand, ground restriction is formally supported by Theorem 10 that states that the application of rule (1a) can be made at the final step of proofs, thus the introduction of blank nodes is not crucial for proofs.

We show in this section that the two above mentioned restrictions considerably simplify the set of inference rules for RDFS. In fact, we consider the set of rules (1b), (2), (3) and (4), and we prove that these rules define a sound and complete system for RDFS entailment under the above mentioned restrictions.

We start by defining the notions of *minimal RDFS triple* and *minimal RDFS graph*, and the system of rules that we use for inference.

Definition 15 (Minimal RDFS) *A* minimal RDFS triple *(*mrdf*–triple for short) is a ground* ρdf*–triple having no* ρdf *vocabulary as subject or object. A* mrdf*–graph is a set of* mrdf*–triples.*

For G and H mrdf*–graphs, define $G \vdash_{\text{mrdf}} H$ iff there is proof of H form G (in the sense of Definition 6) involving solely the rules (1b),* (2)*,* (3) *and* (4)*.*

This refinement is well behaved, indeed using a similar argument as in the proof of Proposition 7 it can be shown:

Proposition 16 *Let G be a* mrdf*–graph and let H be an RDFS graph. Assume that $G \vdash_{\text{mrdf}} H$. Then H is a* mrdf*–graph.*

The following result proves that $\vdash_{\text{mrdf}}$ is sound and complete with respect to the reflexive-relaxed entailment $\models^{\text{nrx}}_{\rho\text{df}}$ (see Definition 12) for mrdf-graphs.

Proposition 17 *Let G and H be* mrdf*–graphs. Then*

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

The proof of the proposition follows from the proof of Theorem 8. Notice that the mrdf fragment considers only ground graphs and that any ground application of rule (5) reduces to the application of rules (2b) plus rule (4), thus, we do not need rule (5) for $\models^{\text{nrx}}_{\rho\text{df}}$. Also notice that, since $\models^{\text{nrx}}_{\rho\text{df}}$ does not force the reflexivity constrains over `sp` and `sc`, we can avoid rules (6) and (7). The details of the proof can be found in the appendix.

The following result (that follows from Corollary 9, Theorem 13, and Proposition 17) states that the normative semantics of RDFS [17] is captured by the deductive system $\vdash_{\text{mrdf}}$ for those mrdf–graphs that do not mention triples of the form $(x, \texttt{sp}, x)$ nor $(x, \texttt{sc}, x)$ for $x \in \mathbf{UL}$.

Corollary 18 *Let $\models$ be the RDFS entailment defined in [17], and let G and H be* mrdf*–graphs. Assume that H does not contain triples of the form $(x, \texttt{sp}, x)$ nor $(x, \texttt{sc}, x)$ for $x \in \mathbf{UL}$. Then*

$$G \models H \text{ iff } G \vdash_{\text{mrdf}} H.$$

Thus, by considering rules (1b), (2)–(4). we have obtained a very simple deductive system that preserves the normative RDFS semantics for a large class of graphs.

4.1. *Complexity*

In this section, we study the complexity of entailment for $\vdash_{\text{mrdf}}$. We prove a lower bound for testing entailment and provide an algorithm that matches this bound. We assume that the data structure used to store RDF graphs is a set of triples, thus, the input of our algorithm is a set.

Let us introduce some notation. For a mrdf–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 $G(\texttt{sp})$ be the directed graph whose vertices are all the elements v which occur as subject or objects in the triples of G, and in which (u, v) is an edge if and only if $(u, \texttt{sp}, v) \in G$. Define $G(\texttt{sc})$ similarly. Let $V_\emptyset$ be the vocabulary that is not RDFS vocabulary.

The naive approach to test the entailment $G \vdash_{\text{mrdf}} H$ in the ground case would be to consider the *closure* of G, that is the graph obtained by adding to G all triples that are derivable from G using rules (2), (3) and (4), and check if H is included in that closure. The following results show that this procedure would take time $\Theta(|H| \cdot |G|^2)$ in the worst case, which is too expensive from a database point of view in the worst case:

Theorem 19 (Size of closure) *Let G be a* mrdf*–graph.*

(i) The size of the closure of G is $\mathcal{O}(|G|^2)$.

(ii) 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 $G(\texttt{sp})$ graph by the usage of rule (2b): it can be shown that this gives at most $|G_\emptyset| \times |G_{\texttt{sp}}|$ triples. For triples having a fixed predicate in ρdf the quadratic

bound is trivial. The lower bound follows from the example below.

Example 20 (Lower bound for the closure) *Consider the graph*

$$\{(a_1, \mathtt{sp}, a_2), \ldots, (a_n, \mathtt{sp}, a_{n+1})\}.$$

The number of triples of the closure of the graph is $\sum_{k=1}^{n} k$ *that is order* $\Omega(n^2)$.

The algorithm in Table 2 presents a much better procedure to check ground entailment in this fragment as next theorem states.

Theorem 21 *Let* (a, b, c) *be a* mrdf*–triple and let* G *be a* mrdf*–graph. Then the algorithm in Table 2 can be used to test the entailment* $G \vdash_{\text{mrdf}} (a, b, c)$ *in time* $\mathcal{O}(|G| \log |G|)$.

The proof of the theorem rests in the following interpolation lemmas that state which specific subgraph of G is relevant when deriving triples containing ρdf vocabulary as predicate, or having no ρdf vocabulary at all. Let us denote by $G|_S$ the subgraph of G *induced* by vocabulary S, i.e. those triples of G having subject, predicate, or object in S.

Lemma 22 *Let* (a, b, c) *be a* mrdf*–triple and* G *a* mrdf*–graph. Assume that* b *does not belong to* ρdf*. Then* $G \vdash_{\text{mrdf}} (a, b, c)$ *iff* $G|_{\{\mathtt{sp}, a, b, c\}} \vdash_{\text{mrdf}} (a, b, c)$.

PROOF. We only need to prove $G|_{\{\mathtt{sp}, a, b, c\}} \vdash_{\text{mrdf}} (a, b, c)$ under the assumption $G \vdash_{\text{mrdf}} (a, b, c)$ (the other direction is trivial). Assume $G \vdash_{\text{mrdf}} (a, b, c)$. Thus a proof sequence $P_1, \ldots P_{k-1}, P_k$ exists with $P_k = \{(a, b, c)\}$.

Since (a, b, c) is ground, there is an instance $\frac{S}{S'}$ of a rule among (2)-(4) with $S' = \{(a, b, c)\}$. Given that $b \notin \rho$df, the only rule that were applied is (2b), with $S = \{(\mathcal{B}, \mathtt{sp}, b), (a, \mathcal{B}, c)\}$. We assume $\mathcal{B} \neq b$ to avoid a trivial and unnecessary application of the rule. Note that by Proposition 16 and the assumption about G, $\mathcal{B} \notin \rho$df.

Thus we need to show that $G|_{\{\mathtt{sp}, a, b, c\}} \vdash_{\text{mrdf}} (\mathcal{B}, \mathtt{sp}, b)$ and $G|_{\{\mathtt{sp}, a, b, c\}} \vdash_{\text{mrdf}} (a, \mathcal{B}, c)$. We can apply the same argument again to $(a, \mathcal{B}, c)$, avoiding trivial applications of rule (2b). For $(\mathcal{B}, \mathtt{sp}, b)$ we only need to observe that any instance $\frac{R}{R'}$ of a rule among (2)-(4) having $R' = \{(\mathcal{B}, \mathtt{sp}, b)\}$ with $\mathcal{B} \neq b$ must be an instance of rule (2a), thus having as premises $R = \{(\mathcal{B}, \mathtt{sp}, \mathcal{C}), (\mathcal{C}, \mathtt{sp}, b)\}$. We can observe that, if $\mathcal{B} = \mathcal{C}$ or $\mathcal{C} = b$, thus the application of the rule can be avoided. In any case only triples of the form $(a, \mathcal{D}, c)$ or $(\mathcal{C}, \mathtt{sp}, \mathcal{E})$, with $\mathcal{C} \neq \mathcal{D}$, are needed. □

The next lemma characterizes mrdf-entailment of triples with `dom` and `range` predicates:

Lemma 23 *Let* G *be a* mrdf*–graph. Let* $a, b \in$ **UBL** *and assume* a, b *do not belong to* ρdf*. Then*

(i) $G \vdash_{\text{mrdf}} (a, \mathtt{dom}, b)$ *iff* $(a, \mathtt{dom}, b) \in G$

(ii) $G \vdash_{\text{mrdf}} (a, \mathtt{range}, b)$ *iff* $(a, \mathtt{range}, b) \in G$

PROOF.

(i) We only need to prove that $(a, \mathtt{dom}, b) \in G$ under the assumption $G \vdash_{\text{mrdf}} (a, \mathtt{dom}, b)$ (the other direction is trivial). So assume $G \vdash_{\text{mrdf}} (a, \mathtt{dom}, b)$. Thus a proof sequence $P_1, \ldots P_{k-1}, P_k$ exists with $P_k = \{(a, \mathtt{dom}, b)\}$. Thus $(a, \mathtt{dom}, b) \in P_j$ for some $j \leq k$. Assume j is the least index with this property. As we only concern with rules (2)-(4), we have that $(a, \mathtt{dom}, b) \in G$ or there exists an instance $\frac{R}{R'}$ of these rules such that $(a, \mathtt{dom}, b) \in R'$. But this happens only for rule (2b), which forces a triple of the form $(\mathcal{A}, \mathtt{sp}, \mathtt{dom}) \in R$. By Proposition 16, this fact contradicts the assumption about G. Therefore $(a, \mathtt{dom}, b) \in G$.

(ii) The proof of $G \vdash_{\text{mrdf}} (a, \mathtt{range}, b)$ iff $(a, \mathtt{range}, b) \in G$ is similar to the previous proof, replacing `dom` by `range`. □

The next lemma characterizes mrdf-entailment of non-reflexive triples with `sc` and `sp` predicates:

Lemma 24 *Let* G *be a* mrdf*–graph. Let* $a, b \in$ **UBL** *and assume* a, b *do not belong to* ρdf *and that* $a \neq b$*. Then*

(i) $G \vdash_{\text{mrdf}} (a, \mathtt{sc}, b)$ *iff* $G|_{\{\mathtt{sc}\}} \vdash_{\text{mrdf}} (a, \mathtt{sc}, b)$.

(ii) $G \vdash_{\text{mrdf}} (a, \mathtt{sp}, b)$ *iff* $G|_{\{\mathtt{sp}\}} \vdash_{\text{mrdf}} (a, \mathtt{sp}, b)$.

PROOF.

(i) We only need to prove that $G|_{\mathtt{sc}} \vdash_{\text{mrdf}} (a, \mathtt{sc}, b)$ under the assumption $G \vdash_{\text{mrdf}} (a, \mathtt{sc}, b)$ (the other direction is trivial). Assume $G \vdash_{\text{mrdf}} (a, \mathtt{sc}, b)$. Thus a proof sequence $P_1, \ldots P_{k-1}, P_k$ exists with $P_k = \{(a, \mathtt{sc}, b)\}$.

As $(a, \mathtt{sc}, b)$ is ground, it may be in G, in which case it is true that $G|_{\mathtt{sc}} \vdash_{\text{mrdf}} (a, \mathtt{sc}, b)$, or it was obtained at some step by the application of an instance of a rule among (2)-(4). In this last case, let i be the step (P_{i-1}, P_i) where $(a, \mathtt{sc}, b)$ was introduced, by means of the instance $\frac{R}{R'}$ of a rule among (2)-(4); we can assume that i is the least index with this property. But as $a \neq b$, the rule must be (3a), with premises $R = \{(a, \mathtt{sc}, \mathcal{C}), (\mathcal{C}, \mathtt{sc}, b)\}$. If $a = \mathcal{C}$

Algorithm (Minimal RDFS Entailment)
Input: G, triple (a, p, b)
1. IF $p \in \{\texttt{dom}, \texttt{range}\}$ THEN check if $(a, p, b) \in G$.
2. IF $p = \texttt{sp}$, $a \neq b$, THEN check if there is a path from a to b in $G(\texttt{sp})$.
3. IF $p = \texttt{sc}$, $a \neq b$, THEN check if there is a path from a to b in $G(\texttt{sc})$.
4. IF $p \notin \rho\text{df}$ THEN check if $(a, p, b) \in G_\emptyset$; if it is not:
   LET $G(\texttt{sp})^*$ be the graph $G(\texttt{sp})$ with the following marks:
   For each $(u, v, w) \in G_\emptyset$, if $v \in G(\texttt{sp})$ then mark v with (u, w).
   IN Check in $G(\texttt{sp})^*$ if there is a path from a vertex marked with (a, b) which reaches p.
5. IF $p = \texttt{type}$ THEN
   LET $G(\texttt{sp})'$ be the graph $G(\texttt{sp})$ with the following marks:
   - For each triple $(u, \texttt{dom}, v) \in G$, mark the vertex u in $G(\texttt{sp})$ with $d(v)$.
   - For each triple $(z, e, y) \in G_\emptyset$, if $e \in G(\texttt{sp})$, mark the node e with $s(z)$.
   LET L_d be the ordered list of elements $d(v)$ such that there is a path from v to b in $G(\texttt{sc})$
   LET L_s be the ordered list of elements $s(z)$ such that either:
   1) in $G(\texttt{sp})'$ there is a path from a node marked $s(z)$ to a node marked with an element in L_d, or
   2) there is $(z, \texttt{type}, v) \in G$, for $d(v) \in L_d$.
   IN Check if $s(a)$ is in L_s.
6. Repeat point 5 symmetrically for `range` instead of `dom`.
   (making the corresponding changes)

Table 2
Algorithm for checking entailment for minimal RDFS.

or $\mathcal{C} = b$, then $(a, \texttt{sc}, b) \in R \subseteq P_{i-1}$ contradicting the assumption about i. Thus $a \neq \mathcal{C}$ and $\mathcal{C} \neq b$. We can now apply the same argument again to $(a, \texttt{sp}, \mathcal{C})$ and $(\mathcal{C}, \texttt{sc}, b)$.

(ii) The argument for $G \vdash_{\text{mrdf}} (a, \texttt{sp}, b)$ iff $G|_{\texttt{sp}} \vdash_{\text{mrdf}} (a, \texttt{sp}, b)$ is similar to the used above for the proof of $G \vdash_{\text{mrdf}} (a, \texttt{sc}, b)$ iff $G|_{\texttt{sc}} \vdash_{\text{mrdf}} (a, \texttt{sc}, b)$, replacing `sp` by `sc` and (3a) by (2a). □

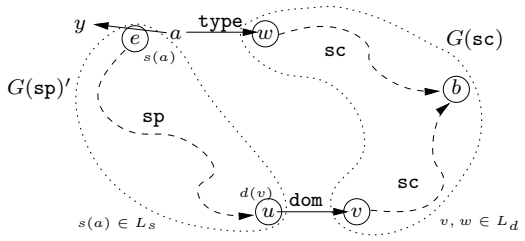

Figure 1. Point 5 of the algorithm for checking entailment for minimal ρdf.

It turns out that `type` is the most complex to deal with, as next lemma shows:

Lemma 25 *Let G and H be* mrdf*–graphs. Assume $G \vdash_{\text{mrdf}} H$. Then:*

(i) If $\texttt{type} \notin \text{voc}(H)$ *then*

$$G|_{\text{voc}(H) \cup \{\texttt{sp}\}} \vdash_{\text{mrdf}} H.$$

(ii) If $\texttt{type} \in \text{voc}(H)$ *then*

$$G|_{\text{voc}(H) \cup \{\texttt{dom}, \texttt{range}, \texttt{sp}, \texttt{sc}\}} \vdash_{\text{mrdf}} H.$$

PROOF. The first statement of the lemma follows directly from Lemmas 22, 23, and 24. The second statement of the lemma follows from the observation that there are two cases for deriving $(x, \texttt{type}, b)$ using rules (2)–(4):

– It can be derived applying rule (3b) from triples $(x, \texttt{type}, a)$ and $(a, \texttt{sc}, b)$, thus, the statement follows recursively from Lemma 24; or
– it can be derived applying rule (4) from triples $(a, \texttt{dom}, b)$ or $(a, \texttt{range}, b)$, plus a triple (x, a, y), thus, the statement follows recursively from Lemma 23 and Lemma 22. □

Correctness and completeness of the algorithm follows from an inspection of the rules and from the previous interpolation lemmas. The algorithm uses the rules in a bottom-up fashion. There are some subtleties in points 4 and 5. Point 4 follows from Lemma 22, which states that the deduction of a triple (a, b, c) with b not belonging to ρdf depends only on the subgraph of G generated by the vocabulary $\{a, b, c, \texttt{sp}\}$. Hence rules (2a) and (2b) are the only relevant: (2a) constructs the graph $G(\texttt{sp})$ and (2b) puts the marks. 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 5 we are checking $(a, \texttt{type}, b)$ (see Figure 1). The crucial observation is that in $G(\texttt{sp})'$, if there is a path from a vertex marked $s(z)$ to a vertex u marked $d(v)$, then $G \vdash_{\text{mrdf}} (z, u, y)$ for some y, and hence $G \vdash_{\text{mrdf}} (z, \texttt{type}, v)$ using rule (4a) (or

rule (4b)). From here it follows that if there is a path in $G(\mathtt{sc})$ from a node v (with $d(v) \in L_d$) to b, then using iteratively the rule (3b) we get $G \vdash_{\text{mrdf}} (z, \mathtt{type}, b)$. Hence a check in the list L_s for $s(a)$ will determine if $G \models (a, \mathtt{type}, b)$ or not.

Now it rest to check if all this can be done efficiently. First, it is no difficult to see that the construction of $G(\mathtt{sp})'$ can be done in time $|G| \log |G|$. The construction of the list L_d takes time $|G| \log |G|$: List all $d(v)$, order them, and then expand the graph $G(\mathtt{sc})$ from b each time looking at L_d to mark the corresponding element. At the end, delete unmarked elements. The construction of L_s takes time $|G| \log |G|$ as well: First list all elements $s(z)$ (no more than $|G|$) in a list L'_s. Then in the directed graph $G(\mathtt{sp})'$, mark green each node marked with an element of L_d. Now recursively do the following: (1) if a node green is marked with an $s(z_0)$, mark that element in L'_s; (2) Expand the green nodes. At the end of the process, delete from L'_s all non-marked elements to get L_s.

The next result shows that the above algorithm cannot be essentially improved, in the sense that, any other algorithm for testing the ground entailment $G \vdash_{\text{mrdf}} t$ would take time proportional to $|G| \log |G|$ in the worst case.

Theorem 26 *Let t be a* mrdf*–triple and let G be a* mrdf*–graph. Then testing $G \vdash_{\text{mrdf}} t$ uses $\Omega(|G| \log |G|)$ comparisons between RDF terms in the worst case.*

PROOF. The bound is obtained by coding the problem of determining whether two sets are disjoint, which is a well known problem that needs $\Omega(n \log n)$ comparisons in the worst case [2]. Given the sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$, construct a graph G 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, we have that $G \vdash_{\text{mrdf}} (x, a_n, y)$ iff $A \cap B \neq \emptyset$. □

Finally, we can state a bound for deciding $G \vdash_{\text{mrdf}} H$.

Corollary 27 *Let G and H be* mrdf*–graphs. Then deciding if $G \vdash_{\text{mrdf}} H$ can be done in time $\mathcal{O}(|H||G| \log |G|)$.*

4.2. *An extension with blank nodes*

For the sake of completeness, we state here the complexity of deduction for graphs including blank nodes. The hardness part of the following complexity result has appeared in several papers in different formulations and for different fragments of RDFS (for example [17], [16], [1], [24], and [9]), that is, it belongs to the *folklore* of RDF.

Theorem 28 (Folklore) *Given (not necessarily ground) graphs G and H, deciding if $G \vdash_{\rho\text{df}} H$ is NP-complete.*

In spite of the above apparently bad complexity behavior, polynomial bounds for the problem can be derived from well-known databases and constraint satisfaction results [6,9,1,14], by considering special forms of interaction between blank nodes and applying Theorem 10.

For the purposed uses of RDFS, the whole Web, polynomial bounds are still not satisfactory. In the next result we provide a tight bound for entailment for a big class of RDFS graphs containing blank nodes, that are likely to occur in practice. Indeed the theorem shows that for that class, testing ρdf entailment in general can be done asymptotically as fast as for the ground case.

Theorem 29 *Let G and H be graphs in minimal RDFS extended by allowing that each triple in H has at most one blank node. Then deciding whether $G \vdash_{\rho\text{df}} H$ can be done in time $\mathcal{O}(|H||G| \log |G|)$.*

PROOF. Let H_0 be the subset of ground triples of H. We already know that checking $G \vdash_{\rho\text{df}} H_0$ can be done in time $\mathcal{O}(|H_0||G| \log |G|)$. For the non-ground part, let $H' = H \setminus H_0$, and assume that k blank nodes occur in H'. For each such blank node X_j $(1 \leq j \leq k)$ in H' consider the graph H_j consisting of all triples of H' in which X_j occurs. Because each triple has at most one blank node, H' is the disjoint union of the H_j for $j = 1, \ldots k$, and, if $G \vdash_{\rho\text{df}} H'$ then the map of the last step in Theorem 10 is as well a disjoint union of the maps μ_j which witness each $G \vdash_{\rho\text{df}} H_j$. Hence, it is enough to check if $G \vdash_{\rho\text{df}} H_j$ for each $j = 1, \ldots, k$.

Now, for each graph H_j with $j = 1, \ldots, k$ do the following: For each triple $t \in H_j$ build a list L_t with all instantiations v of X_j for which $G \vdash_{\rho\text{df}} t[v/X_j]$. Each such list has at most $\text{voc}(G)$ elements, and then each list is of size at most $|G|$. Using the algorithm for ground entailment, it is not difficult to check that this list can be built in time $\mathcal{O}(|G| \log |G|)$. In fact,

for the cases (1) to (3) in the algorithm, the result is trivial. For (4), expand the graph $G(\mathtt{sp})'$ from p looking for nodes marked (x, b) (resp. (a, x)) and each time one is found, include the value x is the list. For (5), and $(X_j, \mathtt{type}, b)$ note that the list L_s gives all the solutions. For (6) and triple of the form $(a, \mathtt{type}, X_j)$, the list L_d should include all nodes reachable from any $d(v)$. Finally, let $L_1, \ldots, L_{|H_j|}$ be the lists obtained. Order each of them, and then, compute the intersection of L_1, L_2, compute the intersection of this list with L_3 and so inductively, and test if the result is non-empty. This process takes time $\mathcal{O}(|H_j||G| \log |G|)$, and then the whole process takes $\mathcal{O}(|H||G| \log |G|)$. □

5. Related work and concluding remarks

The normative semantics of RDFS [17] has two drawbacks: it is formulated in highly non-standard fashion and is complex to read, which makes it difficult to deal with for the common database, logician or even developers. Several works have dealt with the problem of providing a more clean (or standard) semantics for the normative RDFS semantics. Among them, the first formalizations we can mention is Mendelzon et al. [16] which emphasizes database features, the formalization by Marin [20], and the formalization by ter Horst [24], the last two providing a formal semantics for the whole language, but deviating little from the normative approach. A further discussion on the logical foundations of the RDF specification was done by Bruijn et al [9].

There are other approaches to define a semantics for RDFS which definitively depart from the normative approach. Although, the focus in our work is not to provide a standard semantic definition for RDFS, but to study fragments of the normative semantics with a good compromise between expressiveness and computational complexity, some of our results have a close relationship with these works.

In RDFS(FA) [22], the authors are motivated by the problem of RDFS having a non-standard metamodelling architecture. The paper is mostly focused in describing the RDFS(FA) semantics for RDFS, a semantics that can interoperate with conventional first-order languages like DAML+OIL and OWL. The central idea is to have a *fixed-layer architecture* for RDFS, in order to clear possible confusions in the usage of the same term for multiple roles. The authors claim that these confusions may arise in RDFS, for example, when the same *name* is used as standing for an object, a predicate, a class, etc. in a single ontology. RDFS(FA) is a sublanguage of RDFS, and its semantics explicitly divides the set of discourse in *strata* or *layers*. The authors came up with a semantics that has clear differences with the normative RDFS semantics. In our work, although we have different motivations, we tackle the same problem but from a completely different perspective. We keep the normative RDFS semantics and show that, in an fragment of the language that comprises the core functionalities, a standard semantics, equivalent to the normative one, can be developed, and moreover, show that the remaining ones can be safely avoided.

In RDF-F-Logic [25] the authors propose a semantics for RDFS using an extension of the F-Logic formalism [18]. This extension is expressive enough to deal with anonymous resources (blank nodes) and reification. With the F-Logic formalism for RDFS the authors obtain by free an expressive framework for modeling ontologies, and a powerful deductive system. The authors claim that an important advantage of their approach is that F-Logic would gave RDFS a standard (second order) logical semantics. As we have mentioned, instead of extending the expressive power of RDFS, our main interest have been to study fragments with less but enough expressive capabilities for most uses, yet having an efficient reasoning system.

A third group of proposals deal with the relationship/interplay of RDFS and ontology languages, namely OWL and the family of description logics known as DL-Lite. Cuenca Grau [15] proposes RDFS(DL), a sublanguage of the RDFS(FA) discussed above. RDF-Schema(DL) essentially amount to suppress the meta-modelling architecture of RDFS(FA) and takes advantage of developments in description logics, particularly OWL-Lite. The motivation for this design decision is that this meta-modelling feature in RDFS(FA) does not seem to increase the expressive power of the language. A relevant development for the RDFS(DL) proposal is a family of description logics, DL-Lite that was introduced to deal with the complexity concerns while keeping as much expressive power as possible, and improving on some database aspects like querying [4]. The RDFS(DL) approach differs from ours in that its main goal is more directed to compatibility with OWL and upper layers than to design a minimalist sublanguage of RDFS with low complexity directly oriented to database processing.

Concluding Remarks. We presented a streamlined fragment of RDFS which includes the most important and used vocabulary for describing data, avoiding vocabulary and semantics that ideally should be part of the structure of the language. We gave a semantics and a set of rules that captures for this fragment precisely the standard semantics of RDF as defined in the W3C Recommendation.

Based on this fragment, we isolated a minimal system for RDFS. This fragment avoids features that are rarely used such as reflexivity and distinguished vocabulary in object and subject positions. We have presented a very simple deductive system for this fragment, we have proved some complexity bounds for the entailment problem, and we have provided an algorithm to test entailment that is asymptotically optimal. This fragment should work as an essential minimal and efficient floor from where to extend the language in different directions.

We think that the minimal RDFS deductive system that we have presented in this paper, by their simplicity and efficiency, could be of help for developers, designers, and theoreticians that work at the data level in the Semantic Web.

Acknowledgments. The authors would like to thank the anonymous referees for their careful reading of the paper, and for providing many useful comments. The authors were supported by: Gutierrez - Fondecyt grant 1070348; Pérez - Conicyt Ph.D. Scholarship; Gutierrez, Munoz-Venegas, and Pérez - grant P04-067-F from the Millennium Nucleus Center for Web Research.

References

[1] J.-F. Baget, RDF entailment as a graph homomorphism, in: Fourth International Semantic Web Conference, 2005.

[2] M. Ben-Or, Lower bounds for algebraic computation trees, in: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, 1983.

[3] T. Berners-Lee, Principles of design, `http://www.w3.org/ DesignIssues/Principles.html`, personal Notes.

[4] A. Borgida, D. Calvanese, M. Rodriguez-Muro, Explanation in the dl-lite family of description logics, in: Proc. of the 7th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE 2008), 2008.

[5] J. J. Carroll, C. Bizer, P. J. Hayes, P. Stickler, Named graphs, Journal of Web Semantics 3 (4) (2005) 247–267.

[6] V. Dalmau, P. G. Kolaitis, M. Y. Vardi, Constraint satisfaction, bounded treewidth, and finite-variable logics, in: Principles and Practice of Constraint Programming - CP 2002, 2002.

[7] L. M. Dan Brickley (ed.), FOAF Vocabulary Specification, `http://xmlns.com/foaf/0.1/`, 2005 (July 2005).

[8] Dbpedia project, `http://dbpedia.org/`.

[9] J. de Bruijn, E. Franconi, S. Tessaris, Logical reconstruction of RDF and ontology languages, in: Principles and Practice of Semantic Web Reasoning, Third International Workshop, 2005.

[10] J. de Bruijn, S. Heymans, Logical foundations of (e)rdf(s): Complexity and reasoning, in: 6th International Semantic Web Conference, 2007.

[11] R. G. Edit. D. Brickley, RDF vocabulary description language 1.0: RDF schema, W3C recommendation (February 2004).

[12] J. J. C. Edit. G. Klyne, RDF concepts and abstract syntax, W3C recommendation (February 2004).

[13] Gene ontology, `http://www.geneontology.org/`.

[14] G. Gottlob, N. Leone, F. Scarcello, The complexity of acyclic conjunctive queries, J. ACM 48 (3) (2001) 431–498.

[15] B. C. Grau, A possible simplification of the semantic web architecture, in: Proceedings of the 13th international conference on World Wide Web, 2004.

[16] C. Gutiérrez, C. A. Hurtado, A. O. Mendelzon, Foundations of semantic web databases, in: Proceedings of the Twenty-third Symposium on Principles of Database Systems, 2004.

[17] P. Hayes, RDF semantics, W3C recommendation (February 2004).

[18] M. Kifer, G. Lausen, J. Wu, Logical foundations of object-oriented and frame-based languages, J. ACM 42 (4) (1995) 741–843.

[19] Linked data, http://linkeddata.org/.

[20] D. Marin, A formalization of RDF (applications de la logique á la sémantique du web), Tech. rep., École Polytechnique – Universidad de Chile, dept. Computer Science, Universidad de Chile, TR/DCC-2006-8. `http://www.dcc.uchile.cl/cgutierr/ftp/draltan.pdf` (2004).

[21] S. Muñoz, J. Pérez, C. Gutiérrez, Minimal deductive systems for RDF, in: Fourth European Semantic Web Conference, 2007.

[22] J. Z. Pan, I. Horrocks, RDFS(FA) and RDF MT: Two semantics for RDFS, in: Second International Semantic Web Conference, 2003.

[23] E. Prud'hommeaux, A. Seaborne, SPARQL query language for RDF. W3C recommendation, `http://www.w3.org/TR/rdf-sparql-query/` (January 2008).

[24] H. J. ter Horst, Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary, Journal of Web Semantics 3 (2-3) (2005) 79–115.

[25] G. Yang, M. Kifer, Reasoning about anonymous resources and meta statements on the semantic web, Journal on Data Semantics 1 (2003) 69–97.

Appendix A. RDFS definitions

In this and the following section we denote by rdfsV the RDFS vocabulary (as shown in Table A.1). Recall that we assume that the RDFS vocabulary already contains the RDF vocabulary [17].

Definition 30 (cf. [17,20]) *The interpretation* $\mathcal{I} = (Res, Prop, Class, Ext, CExt, Lit, Int)$ *is an* RDFS model *for an RDF graph* G*, denoted by* $\mathcal{I} \models G$*, iff* $\mathcal{I}$ *is an interpretation over vocabulary* rdfsV $\cup$ terms(G) *that satisfies the RDFS axiomatic triples [17] (see Table A.2) and the following semantic conditions:*

(i) 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*.*

(ii) RDF:
- *(a)* $x \in Prop$ *iff* $(x, Int(\texttt{prop})) \in Ext(Int(\texttt{type}))$
- *(b) If* $l \in$ terms(G) *is a well-typed XML literal [17] with lexical form* w*, then* $Int(l)$ *is the XML literal value of* w*,* $Int(l) \in Lit$*, and* $(Int(l), Int(\texttt{xmlLit})) \in Ext(Int(\texttt{type}))$*. Otherwise, if* l *is an ill-typed XML literal [17] then* $Int(l) \notin Lit$ *and* $(Int(l), Int(\texttt{xmlLit})) \notin Ext(Int(\texttt{type}))$*.*

(iii) RDFS Classes:
- *(a)* $x \in Res$ *iff* $x \in CExt(Int(\texttt{res}))$
- *(b)* $x \in Class$ *iff* $x \in CExt(Int(\texttt{class}))$
- *(c)* $x \in Lit$ *iff* $x \in CExt(Int(\texttt{literal}))$

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

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

(vi) RDFS Typing:
- *(a)* $x \in CExt(y)$ *iff* $(x, y) \in Ext(Int(\texttt{type}))$
- *(b) if* $(x, y) \in Ext(Int(\texttt{dom}))$ *and* $(u, v) \in Ext(x)$ *then* $u \in CExt(y)$
- *(c) if* $(x, y) \in Ext(Int(\texttt{range}))$ *and* $(u, v) \in Ext(x)$ *then* $v \in CExt(y)$

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

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

Appendix B. Proofs

Proof of Theorem 5

In the proof of this Theorem we use Definition 30 of Appendix A for RDFS models.

$\Leftarrow$) Assume that $G \models_{\rho\text{df}} H$ and let $\mathcal{I}$ be an RDFS model of G, that is, $\mathcal{I}$ satisfies all the conditions in Definition 30 for G. We need to show that $\mathcal{I}$ is an RDFS model of H.

As $\mathcal{I}$ satisfies conditions (i), (iv), (v), and (vi) of Definition 30, we have that $\mathcal{I}$ interprets every element in ρdf as a property name, and it also satisfies the axiomatic triples

$$(\texttt{dom}, \texttt{dom}, \texttt{prop}),\ (\texttt{dom}, \texttt{range}, \texttt{class}),$$
$$(\texttt{range}, \texttt{dom}, \texttt{prop}),\ (\texttt{range}, \texttt{range}, \texttt{class}),$$
$$\text{and } (\texttt{type}, \texttt{range}, \texttt{class})$$

Thus $\mathcal{I}$ also satisfies all conditions in Definition 4 and so $\mathcal{I}$ is also a ρdf model for G. Now $\mathcal{I}$ is a ρdf model for H that satisfies all the conditions of Definition 30 and then $\mathcal{I}$ is also an RDFS model for H, completing this part of the proof.

$\Rightarrow$) Assume that $G \models H$, and let $\mathcal{I} = (Res, Prop, Class, Ext, CExt, Lit, Int)$ be a ρdf-model of G. We need to show that $\mathcal{I}$ is a ρdf-model of H. To prove this property, we first construct an RDFS model $\mathcal{I}'$ of G from $\mathcal{I}$.

First, we assume that for each $\texttt{e} \in$ rdfsV there is an element $x_{\texttt{e}}$ that is used to interpret $\texttt{e}$ in $\mathcal{I}'$. Further assume that $Int(\texttt{e}) = x_{\texttt{e}}$ for every $\texttt{e} \in \rho$df in $\mathcal{I}$. Since $\mathcal{I}$ is an interpretation under ρdf, it follows that $Int(\texttt{e})$ is not defined for each $\texttt{e} \in$ rdfsV $- \rho$df (the set difference). Now, let Ax be the set of all RDFS axiomatic triples [17] (see Table A.2). With the above observation we construct the interpretation $\mathcal{I}'$ as follows. Let $\mathcal{I}' = (Res', Prop', Class', Ext', CExt', Lit', Int')$ be the interpretation defined on the top of $\mathcal{I}$ by:

- $Res' = Res \cup Prop \cup \{x_{\texttt{e}} \mid \texttt{e} \in \text{rdfsV}\}$.
- $Prop' = Prop \cup \{x_{\texttt{e}} \mid \texttt{e} \in \rho\text{df}\} \cup$
  $\{x_{\texttt{e}} \mid (\texttt{e}, \texttt{type}, \texttt{prop}) \in Ax\} \cup$
  $\{x_{\texttt{e}} \mid (\texttt{e}, \texttt{sp}, y), (z, \texttt{sp}, \texttt{e}), (\texttt{e}, \texttt{dom}, u), \text{ or } (\texttt{e}, \texttt{range}, v) \in Ax\} \cup$
  $\{x \mid (x, y) \in Ext(x_{\texttt{sp}}), (z, x) \in Ext(x_{\texttt{sp}}), (x, u) \in Ext(x_{\texttt{dom}}), \text{ or } (x, v) \in Ext(x_{\texttt{range}})\}$.
- $Class' = Class \cup$
  $\{x_{\texttt{e}} \mid (y, \texttt{type}, \texttt{e}) \in Ax\} \cup$
  $\{x_{\texttt{e}} \mid (\texttt{e}, \texttt{sc}, y),\ (z, \texttt{sc}, \texttt{e}),\ (u, \texttt{dom}, \texttt{e}),\ \text{or } (v, \texttt{range}, \texttt{e}) \in Ax\} \cup$
  $\{x \mid (y, x) \in Ext(x_{\texttt{type}})\} \cup$

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 A.1
RDFS vocabulary [17], with shortcuts in brackets

(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)	(5) **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 A.2
RDFS axiomatic triples [17,20]

$\{x \mid (x,y) \in Ext(x_{\mathtt{sc}}), (z,x) \in Ext(x_{\mathtt{sc}}),$
$(u,x) \in Ext(x_{\mathtt{dom}})$, or $(v,x) \in Ext(x_{\mathtt{range}})\}$.

– $Lit' = Lit$.

– Int' is such that $Int'(\mathtt{e}) = x_{\mathtt{e}}$ for each $\mathtt{e} \in$ rdfsV, and $Int'(x) = Int(x)$ in other case.

– Ext' is an extension function such that:

· $Ext'(x_{\mathtt{type}}) =$
$Ext(x_{\mathtt{type}}) \cup$
$\{(x_{\mathtt{s}}, x_{\mathtt{o}}) \mid (\mathtt{s}, \mathtt{type}, \mathtt{o}) \in Ax\} \cup$
$\{(y, x_{\mathtt{res}}) \mid y \in Res'\} \cup$
$\{(y, x_{\mathtt{prop}}) \mid y \in Prop'\} \cup$
$\{(y, x_{\mathtt{class}}) \mid y \in Class'\} \cup$
$\{(y, x_{\mathtt{literal}}) \mid y \in Lit'\} \cup$
$\{(x,y) \mid x \in Res', (x_{\mathtt{e}}, y) \in Ext(x_{\mathtt{dom}}) \cup$
$Ext(x_{\mathtt{range}})$ with $\mathtt{e} \in \rho\text{df}\}$.

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

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

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

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

· $Ext'(x_{\mathtt{e}}) = \emptyset$ for every $x_{\mathtt{e}} \in Prop'$ such that $\mathtt{e} \in$ rdfsV $- \rho$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 $\mathtt{e} \in \rho$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 RDFS axiomatic triple $(\mathtt{s}, \mathtt{p}, \mathtt{o})$ we have that $\mathtt{p} \in \rho$df. From the construction of $Prop'$, Int', and Ext', we have that $Int'(\mathtt{p}) = x_{\mathtt{p}} \in Prop'$ and $(Int'(\mathtt{s}), Int'(\mathtt{o})) = (x_{\mathtt{s}}, x_{\mathtt{o}}) \in Ext'(x_{\mathtt{p}}) = Ext'(Int'(\mathtt{p}))$ for every RDFS axiomatic triple $(\mathtt{s}, \mathtt{p}, \mathtt{o})$. Thus $\mathcal{I}'$ satisfies all RDFS axiomatic triples.

Now we prove that $\mathcal{I}'$ satisfies all the conditions en Definition 30. First observe that $\mathcal{I}$ satisfies conditions (i), (iv), (v), and (vi) of Definition 30 for G, because $\mathcal{I}$ is a ρdf model for G. Now for $\mathcal{I}'$:

(i) Simple:

(a) For every $\mathtt{e} \in \rho$df we have that $Int'(\mathtt{e}) = x_{\mathtt{e}} = Int(\mathtt{e})$ and $Ext(x_{\mathtt{e}}) \subseteq Ext'(x_{\mathtt{e}})$, 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. Hence for every triple $(s,p,o) \in G$ we have that $(Int'(s), Int'(o)) = (Int(s), Int(o)) \in Ext(Int(p)) \subseteq Ext'(Int(p)) = Ext'(Int'(p))$. Therefore $\mathcal{I}'$ satisfies this condition for G.

(ii) RDF:

(a) We have that G does not mention `prop`, so $\mathcal{I}$ does not interpret `prop` and thus there is no y such that $(y, x_{\mathtt{prop}}) \in Ext(x_{\mathtt{type}})$ in $\mathcal{I}$. By definition of $Ext'(x_{\mathtt{type}})$ in $\mathcal{I}'$, we have that $(y, x_{\mathtt{prop}}) \in Ext'(x_{\mathtt{type}})$ iff $y \in Prop'$, and so $\mathcal{I}'$ satisfies this condition for G.

(b) Since G is a ρdf-graph we have that there is no XML typed literal in G. Thus, by the definition of $Ext'(x_{\mathtt{type}})$ and $CExt'(x_{\mathtt{xmlLit}})$, we have that $\mathcal{I}'$ satisfies this condition for G.

(iii) RDFS Classes:

(a) By the construction of $\mathcal{I}'$ we have $CExt'(x_{\mathtt{res}}) = Res'$.

(b) By the construction of $\mathcal{I}'$ we have $CExt'(x_{\mathtt{class}}) = Class'$.

(c) By the construction of $\mathcal{I}'$ we have $CExt'(x_{\mathtt{literal}}) = Lit'$.

(iv) RDFS Subproperty:

(a) By the construction of $\mathcal{I}'$ we have that $Ext'(x_{\mathtt{sp}})$ is reflexive over $Prop'$. Now, note that the only axiomatic triple that mention `sp` in its predicate position is $(\mathtt{isDefined}, \mathtt{sp}, \mathtt{seeAlso})$. Thus we must only prove that

$$Ext(x_{\mathtt{sp}}) \cup \{(x_{\mathtt{isDefined}}, x_{\mathtt{seeAlso}}), (x_{\_1}, x_{\mathtt{member}}), (x_{\_2}, x_{\mathtt{member}}), \ldots\}$$

is a transitive relation, which is a direct consequence of the fact that $Ext(x_{\mathtt{sp}})$ is transitive

and that G does not mention $\texttt{isDefined}$, nor $\texttt{seeAlso}$, nor $\texttt{\_i}$ for any $\texttt{i}$.

(b) Let $(x,y) \in Ext'(x_{\texttt{sp}}) = Ext(x_{\texttt{sp}}) \cup \{(x_{\texttt{isDefined}}, x_{\texttt{seeAlso}})\} \cup \{(x,x) \mid x \in Prop'\} \cup \{(x_{\texttt{\_1}}, x_{\texttt{member}}), (x_{\texttt{\_2}}, x_{\texttt{member}}), \ldots\}$. For $(x,y) \in Ext(x_{\texttt{sp}})$ the condition holds because $\mathcal{I}$ satisfies this condition. For $(x_{\texttt{isDefined}}, x_{\texttt{seeAlso}})$ we have that $x_{\texttt{isDefined}}, x_{\texttt{seeAlso}} \in Prop'$ and $Ext'(x_{\texttt{isDefined}}) = \emptyset \subseteq Ext'(x_{\texttt{seeAlso}})$. For $(x_{\texttt{\_i}}, x_{\texttt{member}})$ we have that $x_{\texttt{\_i}}, x_{\texttt{member}} \in Prop'$ and $Ext'(x_{\texttt{\_i}}) = \emptyset \subseteq Ext'(x_{\texttt{member}})$. Finally, for $(x,x) \in Ext'(x_{\texttt{sp}})$ we have $x \in Prop'$ and so the condition holds.

(v) RDFS Subclass:

(a) By the construction of $\mathcal{I}'$ we have that $Ext'(x_{\texttt{sc}})$ is reflexive over $Class'$. Note that $(\texttt{c}_3, \texttt{sc}, \texttt{c}_4)$ for every pair of axiomatic triples $(\texttt{c}_1, \texttt{sc}, \texttt{c}_2)$, so we have that $\texttt{c}_2 \neq \texttt{c}_3$, and also that $\texttt{c}_1, \texttt{c}_2, \texttt{c}_3, \texttt{c}_4 \in \text{rdfsV} - \rho\text{df}$ (see Table 2). Consider $(x,y),(y,z) \in Ext'(x_{\texttt{sc}})$: if $x = y$ or $y = z$, we have $(x,z) \in Ext'(x_{\texttt{sc}})$; 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 $(x,y),(y,z) \in Ext(x_{\texttt{sc}})$ and then by transitivity of $Ext(x_{\texttt{sc}})$ we have that $(x,z) \in Ext(x_{\texttt{sc}}) \subseteq Ext'(x_{\texttt{sc}})$. Finally $Ext'(x_{\texttt{sc}})$ is a transitive relation.

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

(vi) RDFS Typing:

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

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

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

ferent to Ext only in elements $x_{\mathtt{e}}$ with $\mathtt{e} \in \rho\mathrm{df}$, so all remaining cases that left to be checked are the ones in which $(u, v) \in Ext'(x_{\mathtt{e}})$ with $\mathtt{e} \in \rho\mathrm{df}$. We consider now all the remaining cases.

Case $x = x_{\mathtt{type}}$: If $(x_{\mathtt{type}}, y) \in Ext'(x_{\mathtt{dom}})$, and $(u, v) \in Ext'(x_{\mathtt{type}})$ we must prove that $u \in CExt'(y)$. First, for $(x_{\mathtt{type}}, y) \in Ext'(x_{\mathtt{dom}})$ we have $(x_{\mathtt{type}}, y) \in Ext(x_{\mathtt{dom}})$ or $y = x_{\mathtt{res}}$ by the axiomatic triple $(\mathtt{type}, \mathtt{dom}, \mathtt{res})$. If $y = x_{\mathtt{res}}$ and $(u, v) \in Ext'(x_{\mathtt{type}})$, the condition holds because $u \in Res' = CExt'(x_{\mathtt{res}}) = CExt'(y)$. Assuming that $(x_{\mathtt{type}}, y) \in Ext(x_{\mathtt{dom}})$, by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and from $u \in Res'$ we obtain $u \in CExt'(y)$.

Case $x = x_{\mathtt{dom}}$: If $(x_{\mathtt{dom}}, y) \in Ext'(x_{\mathtt{dom}})$, and $(u, v) \in Ext'(x_{\mathtt{dom}})$ we must prove that $u \in CExt'(y)$. First, for $(x_{\mathtt{dom}}, y) \in Ext'(x_{\mathtt{dom}})$ we get $(x_{\mathtt{dom}}, y) \in Ext(x_{\mathtt{dom}})$ or $y = x_{\mathtt{prop}}$ by the axiomatic triple $(\mathtt{dom}, \mathtt{dom}, \mathtt{prop})$. If $y = x_{\mathtt{prop}}$ and $(u, v) \in Ext'(x_{\mathtt{dom}})$, we have two cases: if $(u, v) \in \{(x_{\mathtt{s}}, x_{\mathtt{o}}) \mid (\mathtt{s}, \mathtt{dom}, \mathtt{o}) \in Ax\}$, by the construction of $\mathcal{I}'$ we have $u \in Prop' = CExt'(x_{\mathtt{prop}}) = CExt'(y)$; if $(u, v) \in Ext(x_{\mathtt{dom}})$, by the construction of $Prop'$ we have $u \in Prop'$. Now for $(x_{\mathtt{dom}}, y) \in Ext(x_{\mathtt{dom}})$, by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and as $u \in Res'$ we obtain $u \in CExt'(y)$.

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

Case $x = x_{\mathtt{sp}}$: If $(x_{\mathtt{sp}}, y) \in Ext'(x_{\mathtt{dom}})$, and $(u, v) \in Ext'(x_{\mathtt{sp}})$ we must prove that $u \in CExt'(y)$. First, for $(x_{\mathtt{sp}}, y) \in Ext'(x_{\mathtt{dom}})$ we get $(x_{\mathtt{sp}}, y) \in Ext(x_{\mathtt{dom}})$ or $y = x_{\mathtt{prop}}$ by the axiomatic triple $(\mathtt{sp}, \mathtt{dom}, \mathtt{prop})$. If $y = x_{\mathtt{prop}}$ and $(u, v) \in Ext'(x_{\mathtt{sp}})$, we have several cases: if $(u, v) \in \{(x_{\mathtt{s}}, x_{\mathtt{o}}) \mid (\mathtt{s}, \mathtt{sp}, \mathtt{o}) \in Ax\}$, by the construction of $\mathcal{I}'$ we have $u \in Prop' = CExt'(x_{\mathtt{prop}}) = CExt'(y)$; if $(u, v) \in \{(x, x) \mid x \in Prop'\}$, we get $u \in Prop' = CExt'(x_{\mathtt{prop}}) = CExt'(y)$; if $(u, v) = (x_{\mathtt{\_i}}, x_{\mathtt{member}})$ for some $\mathtt{\_i}$, we get we have $u \in Prop'$, because there is an axiomatic triple $(\mathtt{\_i}, \mathtt{type}, \mathtt{prop})$ for every $\mathtt{\_i}$ and by the construction of $Prop'$; and if $(u, v) \in Ext(x_{\mathtt{prop}})$, by the construction of $Prop'$ we have $u \in Prop'$. Now for $(x_{\mathtt{prop}}, y) \in Ext(x_{\mathtt{dom}})$, by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and so, since $u \in Res'$, we obtain $u \in CExt'(y)$.

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

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

(c) Let $(x, y) \in Ext'(x_{\mathtt{range}})$ 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 $(u, v) \in Ext'(x_{\mathtt{e}})$ with $\mathtt{e} \in \rho\mathrm{df}$.

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

Case $x = x_{\mathtt{dom}}$: If $(x_{\mathtt{dom}}, y) \in Ext'(x_{\mathtt{range}})$, and $(u, v) \in Ext'(x_{\mathtt{dom}})$ we must prove that $v \in CExt'(y)$. First, for $(x_{\mathtt{dom}}, y) \in Ext'(x_{\mathtt{range}})$ we get $(x_{\mathtt{dom}}, y) \in Ext(x_{\mathtt{range}})$ or $y = x_{\mathtt{class}}$ by the axiomatic triple $(\mathtt{dom}, \mathtt{range}, \mathtt{class})$. If $y = x_{\mathtt{class}}$ and $(u, v) \in Ext'(x_{\mathtt{dom}})$, we have two cases: if $(u, v) \in \{(x_{\mathtt{s}}, x_{\mathtt{o}}) \mid (\mathtt{s}, \mathtt{range}, \mathtt{o}) \in Ax\}$, by the construction of $\mathcal{I}'$ we have

$v \in Class' = CExt'(x_{\texttt{class}}) = CExt'(y)$; if $(u, v) \in Ext(x_{\texttt{range}})$, by the construction of $Class'$ we have $v \in Class'$. Now if $(x_{\texttt{dom}}, y) \in Ext(x_{\texttt{range}})$, by the construction of $CExt'$ we have that $CExt'(y) = CExt(y) \cup Res'$ and as $v \in Res'$, we obtain $v \in CExt'(y)$.

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

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

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

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

(vii) RDFS Additionals:

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

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

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

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 so $G \models_{\rho\text{df}} H$, completing the proof.

Proof of Theorem 8

First, in the definition of ρdf models for RDF graphs (Definition 4), the only condition that involves the graph being modeled is condition (i) (*Simple*). The other conditions involve the interpretation itself. It follows that, for an interpretation $\mathcal{I}$ that is a ρdf model for a graph G, testing if it is also a ρdf model for a graph H, we only have to test if $\mathcal{I}$ satisfies condition (i) for H, because $\mathcal{I}$ already satisfies all other conditions (it is already a ρdf model for G).

We split the proof of Theorem 8 in two parts. We first prove the following lemma stating the soundness of the set of rules for $\models_{\rho\text{df}}$.

Lemma 31 *Let G and H be graphs that do not mention RDFS vocabulary outside ρdf. Assume $H \to G$, or $H \subseteq G$, or there is an instantiation $\frac{R}{R'}$ of a rule 2–7 such that $R \subseteq G$ and $H = G \cup R'$. Then $G \models_{\rho\text{df}} H$.*

PROOF. Let $\mathcal{I} = (Res, Prop, Class, Ext, CExt, Int)$ be an interpretation such that $\mathcal{I} \models_{\rho\text{df}} G$, i.e. $\mathcal{I}$ satisfies all the conditions in Definition 4. We know that $\mathcal{I}$ satisfies condition (i) for G. 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 $G \models_{\rho\text{df}} H$ when $H \to G$. 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 $\mu(x) \in \mathbf{B}$ we get $Int_A(\mu(x)) = A(\mu(x)) = A'(x) = Int_{A'}(x)$, (2) if $x \in \mathbf{B}$ but $\mu(x) \notin \mathbf{B}$ we get $Int_A(\mu(x)) = Int(\mu(x)) = A'(x) = Int_{A'}(x)$, and (3) if $x \notin \mathbf{B}$, we obtain $\mu(x) = x$ and $Int_A(\mu(x)) = Int(x) = Int_{A'}(x)$. Thus $Int_A(\mu(x)) = Int_{A'}(x)$ for all $x \in \mathbf{UB}$. Let $(s, p, o) \in H$. Hence $(\mu(s), \mu(p), \mu(o)) = (\mu(s), p, \mu(o)) \in G$. By $\mathcal{I} \models_{\rho\text{df}} G$ 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 (i) of Definition 4 for H (with function A') and also satisfies all other conditions of Definition 4. So $\mathcal{I} \models_{\rho\text{df}} H$.

(b) If $H \subseteq G$, we obtain $H \to G$ and so $G \models_{\rho\text{df}} H$.

(2) Subproperty:

(a) Let $(a, \mathtt{sp}, b), (b, \mathtt{sp}, c) \in G$. It follows that $(Int_A(a), Int_A(b)) \in Ext(Int(\mathtt{sp}))$ and $(Int_A(b), Int_A(c)) \in Ext(Int(\mathtt{sp}))$. As $\mathcal{I}$ satisfies condition (iv) we have that $Int_A(a), Int_A(c) \in Prop$. By transitivity $(Int_A(a), Int_A(c)) \in Ext(Int(\mathtt{sp}))$, $\mathcal{I}$ satisfies condition (i) for $G \cup \{(a, \mathtt{sp}, c)\} = H$ and so $\mathcal{I} \models_{\rho\text{df}} H$.

(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 (iv), $Int(b) \in Prop$ and $Ext(Int(a)) \subseteq Ext(Int(b))$ and so $(Int_A(x), Int_A(y)) \in Ext(Int(b))$. Hence $\mathcal{I}$ satisfies condition (i) for $G \cup \{(x, b, y)\} = H$ and thus $\mathcal{I} \models_{\rho\text{df}} H$.

(3) Subclass:

(a) The same proof for rule (2a) works changing $\mathtt{sp}$ by $\mathtt{sc}$ and $Prop$ by $Class$.

(b) Let $(a, \mathtt{sc}, b), (x, \mathtt{type}, a) \in G$. Hence $(Int_A(x), Int_A(a)) \in Ext(Int(\mathtt{type}))$, and $(Int_A(a), Int_A(b)) \in Ext(Int(\mathtt{sc}))$. By condition (vi) we have $Int_A(a) \in Class$ and $Int_A(x) \in CExt(Int_A(a))$. By condition (v), we get $Int_A(b) \in Class$ and $CExt(Int(a)) \subseteq CExt(Int(b))$, so $Int_A(x) \in CExt(Int_A(b))$. By condition (vi) we get $(Int_A(x), Int_A(b)) \in Ext(Int(\mathtt{type}))$. We have that $\mathcal{I}$ satisfies condition (i) for $G \cup \{(x, \mathtt{type}, b)\} = H$ and so $\mathcal{I} \models_{\rho\text{df}} H$.

(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 (vi) we obtain $Int_A(x) \in CExt(Int_A(b))$, and by condition (vi) again we have that $(Int_A(x), Int_A(b)) \in Ext(Int(\mathtt{type}))$. Hence $\mathcal{I}$ satisfies condition (i) for $G \cup \{(x, \mathtt{type}, b)\} = H$ and so $\mathcal{I} \models_{\rho\text{df}} H$.

(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) and condition (iv) we have $Int(c), Int_A(a) \in Prop$ and $Ext(Int(c)) \subseteq Ext(Int_A(a))$, and so from (3) we obtain that $(Int_A(x), Int_A(y)) \in Ext(Int_A(a))$. From this last result, (1), and condition (vi) we obtain that $Int_A(b) \in Class$ and $Int_A(x) \in CExt(Int_A(b))$. Finally applying condition (vi) again we have that $(Int_A(x), Int_A(b)) \in Ext(Int(\mathtt{type}))$, and so $\mathcal{I}$ satisfies condition (i) for $G \cup \{(x, \mathtt{type}, b)\} = H$ and then $\mathcal{I} \models_{\rho\text{df}} H$.

(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. Thus $Int(a) \in Prop$, and by the reflexivity of $Ext(Int(\mathtt{sp}))$ over $Prop$, we obtain that $(Int(a), Int(a)) \in Ext(Int(\mathtt{sp}))$, and so $\mathcal{I}$ satisfies condition (i) for $G \cup \{(a, \mathtt{sp}, a)\} = H$. Thus $\mathcal{I} \models_{\rho\text{df}} H$.

(b) Let $(a, \mathtt{sp}, b) \in G$. By condition (iv) 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 $p \in \rho\text{df}$ are satisfied by any interpretation (see Proposition 11), and so $\mathcal{I} \models_{\rho\text{df}} G \cup \{(p, \mathtt{sp}, p)\}$ for every $p \in \rho\text{df}$.

(d) Let $(a, p, x) \in G$ with $p \in \{\mathtt{dom}, \mathtt{range}\}$. By the new conditions of ρdf 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 (v) we have that $Int(a), Int(b) \in Class$ and so, by the reflexivity of $Ext(Int(\mathtt{sc}))$ over $Class$, we obtain that

$$(Int_A(a), Int_A(a)), (Int_A(b), Int_A(b)) \in Ext(Int(\mathtt{sc}))$$

Thus $\mathcal{I}$ satisfies condition (i) for $G \cup \{(a, \mathtt{sc}, a), (b, \mathtt{sc}, b)\} = H$ and so $\mathcal{I} \models_{\rho\text{df}} H$.

(b) Assume $p \in \{\mathtt{dom}, \mathtt{range}, \mathtt{type}\}$. Let $(x, p, a) \in G$. By the new condition of ρdf 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 $G \models_{\rho\text{df}} H$. □

To state the completeness of the set of rules, we must introduce the following notion of ρdf *closure*

of a graph. Define the graph ρdf-cl(G) as the *closure* of G under the application of rules (2) to (7). Note that ρdf-cl(G) is an RDF graph over terms(G)$\cup\rho$df, that is a superset of G, and that is obtained after a finite number of application of rules.

Lemma 32 *Given a graph G that do not mention RDFS vocabulary outside ρdf, define the interpretation $\mathcal{I}_G = (Res, Prop, Class, Ext, CExt, Lit, Int)$ such that:*

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

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

PROOF. We must show that $\mathcal{I}_G$ satisfies all the conditions of Definition 4 for G.

(i) Simple:

(a) First note that by construction of ρdf-cl(G), $Res = \text{terms}(\rho\text{df-cl}(G)) = \text{terms}(G) \cup \rho\text{df}$. Take the function $A : \mathbf{B} \to \text{terms}(G) \cup \rho\text{df}$ 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 $G \subseteq \rho\text{df-cl}(G)$. We also have that $(Int_A(s), Int_A(o)) = (s,o) \in Ext(Int(p)) = Ext(p)$ by the definition of Ext because $G \subseteq \rho\text{df-cl}(G)$. Finally we have that $\mathcal{I}_G$ satisfies condition (i) for G.

(ii) Subproperty:

(a) Let $(a,b),(b,c) \in Ext(Int(\texttt{sp})) = Ext(\texttt{sp})$. By construction of $\mathcal{I}_G$ (because $\texttt{sp} \notin \mathbf{B}$) we have that $(a, \texttt{sp}, b)$, $(b, \texttt{sp}, c) \in \rho\text{df-cl}(G)$. thus $a, b, c \in Prop$. As ρdf-cl(G) is closed under application of rule (2a), we have that $(a, \texttt{sp}, c) \in \rho\text{df-cl}(G)$ and so $(a,c) \in Ext(\texttt{sp}) = Ext(Int(\texttt{sp}))$. We conclude that $Ext(Int(\texttt{sp}))$ is a transitive relation. We must show that $Ext(Int(\texttt{sp}))$ is also reflexive over $Prop$. Let $a \in Prop$. By the definition of $Prop$ we have three cases: (1) $(x,a,y) \in \rho\text{df-cl}(G)$; (2) $a \in \rho\text{df}$; (3) $(a, \texttt{sp}, b)$, $(b, \texttt{sp}, a)$, $(a, \texttt{dom}, x)$, or $(a, \texttt{range}, x) \in \rho\text{df-cl}(G)$. Because ρdf-cl(G) is closed under application of rules (6) we obtain that in any case $(a, \texttt{sp}, a) \in \rho\text{df-cl}(G)$ and so $(a,a) \in Ext(\texttt{sp}) = Ext(Int(\texttt{sp}))$ and hence $Ext(Int(\texttt{sp}))$ is reflexive over $Prop$.

(b) Let $(a,b) \in Ext(Int(\texttt{sp})) = Ext(\texttt{sp})$, then by construction of $\mathcal{I}_G$ we have that $(a, \texttt{sp}, b) \in \rho\text{df-cl}(G)$, 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. Assuming that $(x,y) \in Ext(a)$, we have two cases:
- If $a \in \mathbf{U}$, by definition $(x,a,y) \in \rho\text{df-cl}(G)$. Now, if $b \in \mathbf{U}$ because ρdf-cl(G) is closed under application of rule (2b) we have that $(x,b,y) \in \rho\text{df-cl}(G)$ and then $(x,y) \in Ext(b)$. If $b \in \mathbf{B}$, as $(a, \texttt{sp}, b), (x,a,y) \in \rho\text{df-cl}(G)$ 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 $(a', \texttt{sp}, a)$, $(x, a', y) \in \rho\text{df-cl}(G)$. As ρdf-cl(G) is closed under application of rule (2a), we have $(a', \texttt{sp}, b) \in \rho\text{df-cl}(G)$. If $b \in \mathbf{U}$, as $(a', \texttt{sp}, b), (x, a', y) \in \rho\text{df-cl}(G)$ and ρdf-cl(G) is closed under application of rule (2b), we have that $(x,b,y) \in \rho\text{df-cl}(G)$ and then $(x,y) \in Ext(b)$. If $b \in \mathbf{B}$, as $(a', \texttt{sp}, b), (x, a', y) \in \rho\text{df-cl}(G)$ 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 so $Ext(a) \subseteq Ext(b)$.

(iii) Subclass:

(a) Let $(a,b),(b,c) \in Ext(Int(\texttt{sc})) = Ext(\texttt{sc})$. By the construction of $\mathcal{I}_G$ we have that $(a, \texttt{sc}, b)$, $(b, \texttt{sc}, c) \in \rho\text{df-cl}(G)$. Hence $a, b, c \in Class$. As ρdf-cl(G) is closed under application of rule (3a), we have that $(a, \texttt{sc}, c) \in \rho\text{df-cl}(G)$ and so $(a,c) \in Ext(\texttt{sc}) = Ext(Int(\texttt{sc}))$. We conclude that $Ext(Int(\texttt{sc}))$ is a transitive relation. We must show that $Ext(Int(\texttt{sc}))$ is also reflexive over $Class$. Let $a \in Class$, by the definition of $Class$ we have two cases: (1) $(x, \texttt{type}, a) \in \rho\text{df-cl}(G)$; (2) $(a, \texttt{sc}, b)$,

$(b, \texttt{sc}, a)$, $(x, \texttt{dom}, a)$, or $(x, \texttt{range}, a) \in \rho$df-cl$(G)$. Because ρdf-cl(G) is closed under application of rules (7) we obtain that in any case $(a, \texttt{sc}, a) \in \rho$df-cl$(G)$, hence $(a, a) \in Ext(\texttt{sc}) = Ext(Int(\texttt{sc}))$ and so $Ext(Int(\texttt{sc}))$ is reflexive over $Class$.

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

(iv) Typing I:

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

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

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

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

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

(v) 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 4 for G. Therefore $\mathcal{I}_G \models_{\rho\text{df}} G$. □

Lemma 33 *Let G, H be RDF graphs that do not mention RDFS vocabulary outside* ρdf. *If $G \models_{\rho\text{df}} H$ then there is a map $H \to \rho$df-cl(G).*

PROOF. Consider the interpretation,

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

as defined in Lemma 32. thus $\mathcal{I}_G \models_{\rho\text{df}} G$ and as $G \models_{\rho\text{df}} H$, we have $\mathcal{I}_G \models_{\rho\text{df}} H$. We know that $\mathcal{I}_G$ satisfies condition (i) (*Simple*) for H, hence there exists a function $A : B \to \text{terms}(G) \cup \rho\text{df}$ 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 $Ext(Int(p)) = Ext(p) = \{(s, o) \mid (s, p, o) \in \rho\text{df-cl}(G)\}$. Finally, since $(Int_A(s), Int_A(o)) \in Ext(Int(p))$, we have that $(Int_A(s), Int_A(p), Int_A(o)) \in \rho$df-cl$(G)$ for each $(s, p, o) \in H$ Thus $Int_A : H \to \rho$df-cl(G) is a map such that $Int_A(H) \subseteq \rho$df-cl(G), that is, a map $H \to \rho$df-cl(G). □

From the above lemma it follows that since $G \models_{\rho\text{df}} H$, then H can be obtained from ρdf-cl(G) by using rule (1), and thus, since $G \vdash_{\rho\text{df}} \rho$df-cl$(G)$, it holds that $G \vdash_{\rho\text{df}} H$. Therefore Theorem 8 follows directly from Lemmas 31 and 33.

Proof of Theorem 10

It follows directly from Lemma 33 and the fact that $G \vdash_{\rho\text{df}} \rho$df-cl$(G)$ for every G.

Proof of Proposition 17

As in the proof of Theorem 8, we split the proof in two parts. We first prove the following lemma stating the soundness of rules (1b), (2), (3) and (4) for $\models^{\text{nrx}}_{\rho\text{df}}$.

Lemma 34 *Let G and H be* mrdf*–graphs.If $H \subseteq G$, or if there is an instantiation $\frac{R}{R'}$ of a rule* (2), (3) *and* (4) *such that $R \subseteq G$ and $H = G \cup R'$, then $G \models^{\text{nrx}}_{\rho\text{df}} H$.*

PROOF. Follows from the proof of Lemma 31 in Appendix B as the simple observation that reflexivity of the interpretations of `sp` and `sc` are necessary only for proving that rules (6) and (7) are sound, which are not part of the rules in the statement of the lemma. □

Similarly as we define ρdf-cl(G), define nrx-ρdf-cl(G) but using only rules from (2) to (4). We now have the following Lemma.

Lemma 35 *For a* mrdf*–graph G, consider the interpretation $\mathcal{I}_G$ as in Lemma 32 but using* nrx-ρdf-cl(G) *instead of* ρdf-cl(G). *Then $\mathcal{I}_G \models^{\text{nrx}}_{\rho\text{df}} G$.*

PROOF. Follows from the simple observation that, in the proof of Lemma 32, only rules (6) and (7) are needed to show the reflexivity of the interpretations of `sp` and of `sc`. □

Proposition 17 follows from Lemmas 34 and 35.