I1-D3-2

Extended RDF as a Semantic Foundation of Rule Markup Languages

Project title:	Reasoning on the Web with Rules and Semantics
Project acronym:	REWERSE
Project number:	IST-2004-506779
Project instrument:	EU FP6 Network of Excellence (NoE)
Project thematic priority:	Priority 2: Information Society Technologies (IST)
Document type:	D (deliverable)
Nature of document:	R (report)
Dissemination level:	PU (public)
Document number:	IST506779/Heraklion/I1-D3-2/D/PU/b0
Responsible editors:	Anastasia Analyti, Gerd Wagner
Reviewers:	Tim Furche
Contributing participants:	Heraklion, Eindhoven/Cottbus, Lisbon
Contributing workpackages:	I1
Contractual date of deliverable:	31 August 2005
Actual submission date:	13 October 2005

Abstract

Ontologies and automated reasoning are the building blocks of the Semantic Web initiative. Derivation rules can be included in an ontology to define derived concepts based on base concepts. For example, rules allow to define the extension of a class or property based on a complex relation between the extensions of the same or other classes and properties. On the other hand, the inclusion of negative information both in the form of negation-as-failure and explicit negative information is also needed to enable various forms of reasoning. In this paper, we extend RDF graphs with weak and strong negation, as well as derivation rules. The *ERDF stable model semantics* of the extended framework (*Extended RDF*) is defined, extending RDF(S) semantics. A distinctive feature of our theory, which is based on partial logic, is that both truth and falsity extensions of properties and classes are considered, allowing for truth value gaps. Our framework supports both closed-world and open-world reasoning through the explicit representation of the particular closed-world assumptions and the ERDF ontological categories of total properties and total classes.

Keyword List

RDF, negation, rules, closed-world reasoning

Extended RDF as a Semantic Foundation of Rule Markup Languages

Anastasia Analyti[1], Grigoris Antoniou[1,2],
Carlos Viegas Damásio[3], and Gerd Wagner[4]

[1] Institute of Computer Science, FORTH-ICS, Greece
[2] Department of Computer Science, University of Crete, Greece
[3] Centro de Inteligência Artificial, Universidade Nova de Lisboa, Caparica, Portugal
[4] Institute of Informatics, Brandenburg University of Technology at Cottbus, Germany
analyti@ics.forth.gr, antoniou@ics.forth.gr,
cd@di.fct.unl.pt, G.Wagner@tu-cottbus.de

Abstract. Ontologies and automated reasoning are the building blocks of the Semantic Web initiative. Derivation rules can be included in an ontology to define derived concepts based on base concepts. For example, rules allow to define the extension of a class or property based on a complex relation between the extensions of the same or other classes and properties. On the other hand, the inclusion of negative information both in the form of negation-as-failure and explicit negative information is also needed to enable various forms of reasoning. In this paper, we extend RDF graphs with weak and strong negation, as well as derivation rules. The *ERDF stable model semantics* of the extended framework (*Extended RDF*) is defined, extending RDF(S) semantics. A distinctive feature of our theory, which is based on partial logic, is that both truth and falsity extensions of properties and classes are considered, allowing for truth value gaps. Our framework supports both closed-world and open-world reasoning through the explicit representation of the particular closed-world assumptions and the ERDF ontological categories of total properties and total classes.
Keywords: Extended RDF ontologies, negation, rules, semantics.

1 Introduction

The idea of the Semantic Web is to describe the meaning of web data in a way suitable for automated reasoning. This means that descriptive data (meta-data) in machine readable form are to be stored on the web and used for reasoning. Due to its distributed and world-wide nature, the Web creates new problems for knowledge representation research. In [7], the following fundamental theoretical problems have been identified: negation and contradictions, open-world versus closed-world assumptions, and rule systems for the Semantic Web. For the time being, the first two issues have been circumvented by discarding the facilities to introduce them, namely negation and closed-world assumptions. Though the web ontology language OWL [29], which is based on description logic (DL), includes a form of classical negation through class complements, this form is limited. This is because, to achieve decidability, classes are formed based on specific class constructors and negation on properties is not considered. Rules constitute the next layer over the ontology languages of the Semantic Web and, in contrast to DL, allow arbitrary interaction of variables in the body of the rules. The widely recognized need of having rules in the Semantic Web [22, 34] has restarted the discussion of the fundamentals of closed-world reasoning and the appropriate mechanisms to implement it in rule systems, such as the computational concept of *negation-as-failure*.

The RDF(S)[5] recommendation [18] provides the basic constructs for defining web ontologies and a solid ground to discuss the above issues. RDF(S) is a special predicate logical language that is restricted to existentially quantified conjunctions of atomic formulas, involving binary predicates only. Due to its purpose, RDF(S) has a number of special features that distinguish it from traditional logic languages:

1. It uses a special jargon, where the things of the universe of discourse are called *resources*, types are called *classes*, and binary predicates are called *properties*. Like binary relations in set theory, properties have a *domain* and a *range*. Resources are classified with the help of the property $rdf{:}type$ (for stating that a resource is of type c, where c is a class).
2. It distinguishes a special sort of resources, called *literal values*, which are denotations of lexical strings.
3. Properties are resources, that is, properties are also elements of the universe of discourse. Consequently, it is possible to state properties of properties, i.e. make statements about predicates.
4. All resources, except anonymous ones and literal values, are named with the help of a globally unique reference schema, called Uniform Resource Identifier (URI), that has been developed for the Web.
5. RDF(S) comes with a non-standard model-theoretic semantics developed by Pat Hayes on the basis of an idea of Christopher Menzel, which allows self-application without violating the axiom of foundation. An example of this is the provable sentence stating that $rdfs{:}Class$, the class of all classes, is an instance of itself.

The predefined vocabulary of RDF comes in two layers:

1. The basic RDF layer, which includes the terms $rdf{:}type$ and $rdf{:}Property$.
2. The RDF Schema (RDFS) layer, which includes the terms: $rdfs{:}Resource$, $rdfs{:}Literal$, $rdfs{:}Class$, $rdfs{:}Datatype$, $rdfs{:}domain$, $rdfs{:}range$, $rdfs{:}subClassOf$, and $rdfs{:}subPropertyOf$.

However, RDF(S) does not support negation and rules. In [39], it was argued that a database, as a knowledge representation system, needs two kinds of negation, namely *weak negation* $\sim$ (expressing negation-as-failure or non-truth) and *strong negation* $\neg$ (expressing explicit negative information or falsity) to be able to deal with partial information. In [40], this point was made for the Semantic Web as a framework for knowledge representation in general. In the present paper we make the same point for the Semantic Web language RDF and show how it can be extended to accommodate the two negations of partial logic [19], as well as derivation rules. We call the extended language *Extended RDF* and denote it by *ERDF*. The model-theoretic semantics of ERDF, called *ERDF stable model semantics*, is developed based on partial logic [19].

[5] RDF(S) stands for *Resource Description Framework (Schema).*

In partial logic, relating strong and weak negation at the interpretation level allows to distinguish four categories of properties and classes. *Partial properties* are properties p that may have truth-value gaps and truth-value clashes, that is $p(x, y)$ is possibly neither true nor false, or both true and false. *Total properties* are properties p that satisfy *totalness*, that is $p(x, y)$ is true or false (but possibly both). *Coherent properties* are properties p that satisfy *coherence*, that is $p(x, y)$ cannot be both true and false. *Classical properties* are total and coherent properties. For classical properties p, the *classical logic law* applies: $p(x, y)$ is either true or false. Partial, total, coherent, and classical classes c are defined similarly, by replacing $p(x, y)$ by $rdf{:}type(x, c)$.

Partial logic allows also to distinguish between predicates (i.e. classes and properties) that are completely represented in a knowledge base and those that are not. The classification if a predicate is completely represented or not is up to the owner of the knowledge base: the owner must know for which predicates there is complete information and for which there is not. Clearly, in the case of a completely represented (*closed*) predicate p, non-truth (as shown by negation-as-failure) implies falsity, and the underlying *completeness declaration* has also been called *Closed-World Assumption (CWA)* in the AI literature. Semantically, a completeness declaration for a predicate p implies that p is total and, hence, the class of closed predicates is a subclass of the class of total predicates.

However, in this paper we do not consider completeness declarations, but a somewhat weaker variant, which takes the form of default rules and which we call *completeness assumptions*. Such a completeness assumption for *closing* a partial property p by default may be expressed in ERDF by means of the rule $\neg p(?x, ?y) \leftarrow \sim p(?x, ?y)$ and for a partial class c, by means of $\neg rdf{:}type(?x, c) \leftarrow \sim rdf{:}type(?x, c))$. In the case of a total predicate p, such a default rule is not applicable because the implicit LEM-disjunctions $p(c) \vee \neg p(c)$ prevent the preferential entailment of $\sim p(c)$. That is, in the case of such open total (and also in the case of open partial) predicates, explicit negative information has to be supplied along with ordinary (positive) information for allowing to infer negated statements.

Unfortunately, neither classical logic nor Prolog supports this distinction between *closed* and *open* predicates. Classical logic supports only open-world reasoning. On the contrary, Prolog supports only closed-world reasoning, as *negation-as-failure* is the only negation mechanism supported. For arguments in favor of the combination of closed and open world reasoning in the same framework, see [3].

Specifically, in this paper:

1. We extend RDF graphs to ERDF graphs with the inclusion of strong negation, and then to ERDF ontologies (or ERDF knowledge bases) with the inclusion of general derivation rules. ERDF graphs allow to express existential positive and negative information, whereas general derivation rules allow inferences based on formulas built using the connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$ and the quantifiers $\forall$, $\exists$.
2. We extend the vocabulary of RDF(S) with the terms $erdf{:}TotalProperty$ and $erdf{:}TotalClass$, representing metaclasses of total properties and total classes, on which the open-world assumption applies.
3. We extend RDFS interpretations to ERDF interpretations including both truth and falsity extensions for properties and classes. Then, we define *co-*

herent ERDF interpretations by imposing coherence on all properties. In the developed model-theoretic semantics of ERDF, we consider only coherent ERDF interpretations. Thus, total properties and classes become synonymous to classical properties and classes.

4. We extend RDF graphs to ERDF formulas that are built from positive triples using the connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$ and the quantifiers $\forall$, $\exists$. Then, we define ERDF entailment between two ERDF formulas, extending RDFS entailment between RDF graphs.
5. We define the ERDF models, Herbrand interpretations, minimal Herbrand models, and stable models of ERDF ontologies. We show that stable model entailment on ERDF ontologies extends ERDF entailment on ERDF graphs, and thus it also extends RDFS entailment on RDF graphs.
6. We show that if all properties are total, classical (boolean) Herbrand model reasoning and stable model reasoning coincide. In this case, we make an open-world assumption for all properties and classes.

A distinctive feature of the developed semantics with respect to [19] is that properties and classes are declared as total on a selective basis, by extending RDF(S) with new built-in classes and providing support for the respective ontological categories. In contrast, in [19], the choice of partial or total should be taken for the complete set of predicates. Thus, the approach presented here is, in this respect, more flexible and general.

The rest of the paper is organized as follows: In Section 2, we present the truth tables of partial logic for weak and strong negation. In Section 3, we extend RDF graphs to ERDF graphs and ERDF formulas. Section 4 defines ERDF interpretations and ERDF entailment. We show that ERDF entailment extends RDFS entailment. In Section 5, we define ERDF ontologies and the Herbrand models of an ERDF ontology. In Section 6, we define the stable models of an ERDF ontology and show that stable model entailment extends RDFS entailment. Section 7 shows that the developed ERDF model theory can be seen as a Tarski-style model theory. Section 8 reviews related work and Section 9 concludes the paper, including future work. The main definitions of RDF(S) semantics are reviewed in Appendix A. Appendix B includes the proofs of the Propositions, presented in the paper.

2 Partial logic semantics for weak and strong negation

In natural language, there are (at least) two kinds of negation: a weak negation expressing non-truth (in the sense of "she doesn't like snow" or "he doesn't trust you"), and a strong negation expressing explicit falsity (in the sense of "she dislikes snow" or "he distrusts you"). Notice that the classical logic *law of the excluded middle* holds only for the weak negation (either "she likes snow" or "she doesn't like snow"), but not for the strong negation: it does not hold that "he trusts you" or "he distrusts you"; he may be neutral and neither trust nor distrust you.

A number of knowledge representation formalisms and systems (see, e.g., [17, 39, 25, 1, 35, 9]) follow this distinction between weak and strong negation in natural language. However, many of them do not come with a Tarski-style model-theoretic semantics.

Classical (two-valued) logic cannot account for two kinds of negation because two-valued (Boolean) truth functions do not allow to define more than one negation. The simplest generalization of classical logic that is able to account for two kinds of negation is *partial logic* [19], which gives up the classical bivalence principle, that is that a statement is either true or false. Partial logic supports two kinds of negation, namely *weak negation* ($\sim$), expressing non-truth, and *strong negation* ($\neg$), expressing falsity, based on the notion of partial interpretation. Specifically, let I be a partial interpretation. A literal[6] of a partial predicate is (a) *true* according to I if I satisfies L ($I \models L$), (b) *not-true* if I doesn't satisfy L ($I \not\models L$), (c) $false$ if I satisfies $\neg L$ ($I \models \neg L$), and (d) $undefined$ or *unknown* if L is neither *true* nor $false$. Note that a *not-true* literal is either $false$ or $undefined$.

In partial logic, it holds $I \models {\sim}L$ iff $I \not\models L$. Additionally, the double negation forms $\neg\neg L$ and $\neg \sim L$ collapse to L, while the double negation form $\sim \neg L$ does not collapse: not disliking snow does not amount to liking snow.

Literals of partial predicates are either *true* or *not-true*. However, a *not-true* literal of a partial predicate is not necessarily $false$. Moreover, a literal can be both *true* and $false$, allowing for inconsistencies. Thus, in the general case, weak and strong negation are unrelated, as it is shown in the following satisfaction table for partial predicates ($\bar{c}$ denotes a sequence of constants $c_1, ..., c_n$, where n is the degree of predicate p). Note that if ${\sim}p(\bar{c})$ is satisfied by a partial interpretation I then $\neg p(\bar{c})$ might be satisfied or not. Similar is the case if ${\sim}p(\bar{c})$ is not satisfied by a partial interpretation I. The truth table for partial predicates is also given below. In the truth table, t indicates that the literal is *true* but not $false$, f indicates that the literal is $false$ but not *true*, u indicates that the literal is $undefined$, and b indicates that the literal is *both true* and $false$, according to a partial interpretation I.

Satisfaction Table **partial predicates**		
$p(\bar{c})$	${\sim}p(\bar{c})$	$\neg p(\bar{c})$
$satisfies$	*doesn't satisfy*	*any*
doesn't satisfy	$satisfies$	*any*

Truth Table **partial predicates**		
$p(\bar{c})$	${\sim}p(\bar{c})$	$\neg p(\bar{c})$
t	f	f
f	t	t
u	t	u
b	f	b

Relating weak and strong negation results in special classes of predicates, namely $total$, *coherent*, and $classical$. Total predicates are partial predicates, for which non-truth implies falsity. Thus, an atom of a total predicate is true or false (but possibly both). Specifically, interpretations of a total predicate p should satisfy the axiom $p(\bar{x}) \vee \neg p(\bar{x})$ or, equivalently, the axiom ${\sim}p(\bar{x}) \supset \neg p(\bar{x})$ ($totalness$). The satisfaction and truth tables for total predicates (according to a partial interpretation I) are modified as follows:

[6] A literal is an atom, the weak negation of an atom, or the strong negation of an atom.

Satisfaction Table **total predicates**		
$p(\bar{c})$	$\sim p(\bar{c})$	$\neg p(\bar{c})$
satisfies	*doesn't satisfy*	*any*
doesn't satisfy	*satisfies*	*satisfies*

Truth Table **total predicates**		
$p(\bar{c})$	$\sim p(\bar{c})$	$\neg p(\bar{c})$
t	f	f
f	t	t
b	f	b

Note that the truth table for total predicates is a subset of the truth table for partial predicates, as a literal of a total predicate can never be $undefined$. For example, in the case of a total predicate, such as *authorOf*, we have the relationship that weak negation implies strong negation:

if $I \models \sim authorOf(John, \text{"}Logic\text{"})$ then $I \models \neg authorOf(John, \text{"}Logic\text{"})$,
and equivalently,
if $I \models \sim \neg authorOf(John, \text{"}Logic\text{"})$ then $I \models authorOf(John, \text{"}Logic\text{"})$

Coherent predicates are partial predicates whose atoms cannot be both $true$ and $false$, enforcing selective consistency. Specifically, interpretations of a coherent predicate p should satisfy the axiom $\sim p(\bar{x}) \vee \sim \neg p(\bar{x})$ or, equivalently, the axiom $\neg p(\bar{x}) \supset \sim p(\bar{x})$ (*coherence*). The satisfaction and truth tables for coherent predicates (according to a partial interpretation I) are modified as follows:

Satisfaction table **Coherent predicates**		
$p(\bar{c})$	$\sim p(\bar{c})$	$\neg p(\bar{c})$
satisfies	*doesn't satisfy*	$doesn'tsatisfy$
doesn't satisfy	*satisfies*	*any*

Truth Table **Coherent predicates**		
$p(\bar{c})$	$\sim p(\bar{c})$	$\neg p(\bar{c})$
t	f	f
f	t	t
u	t	u

Note that the truth table for coherent predicates is a subset of the truth table for partial predicates, as a literal of a coherent predicate can never be both $true$ and $false$. For example, in the case of a coherent predicate, such as *killed*, we have the relationship that strong negation implies weak negation:

if $I \models \neg killed(John, Peter)$ then $I \models \sim killed(John, Peter)$,
and equivalently,
if $I \models killed(John, Peter)$ then $I \models \sim \neg killed(John, Peter)$

Classical predicates are both total and coherent predicates. Thus, literals of classical predicates are either $true$ or $false$, as in classical logic. The satisfaction and truth tables for coherent predicates are modified as follows:

Satisfaction Table **classical predicates**		
$p(\bar{c})$	$\sim p(\bar{c})$	$\neg p(\bar{c})$
satisfies	*doesn't satisfy*	*doesn't satisfy*
doesn't satisfy	*satisfies*	*satisfies*

Truth Table **classical predicates**		
$p(\bar{c})$	$\sim p(\bar{c})$	$\neg p(\bar{c})$
t	f	f
f	t	t

Note that weak and strong negation for classical predicates collapse. Thus, the satisfaction and truth tables for classical predicates coincide with these of classical logic. For example, in the case of a classical predicate, such as being an odd number, non-truth and falsity are equivalent:

$$I \models \neg odd(x) \text{ iff } I \not\models odd(x) \text{ iff } I \models \sim odd(x)$$
and equivalently,
$$I \models odd(x) \text{ iff } I \not\models \neg odd(x) \text{ iff } I \models \sim \neg odd(x)$$

Thus, classical logic can be viewed as the degenerate case of partial logic when all predicates are total.

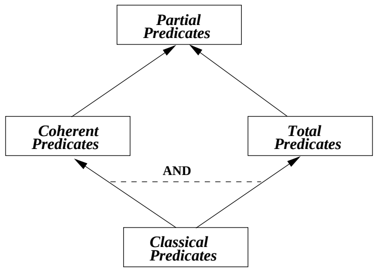

Fig. 1. The subsumption hierarchy of predicate categories of Partial Logic

The subsumption hierarchy of predicate categories of partial logic is given in Figure 1. Note that if all predicates are coherent, the categories of total and classical predicates coincide.

In [19], the (partial logic) stable model semantics of a set of general derivation rules[7] are defined as a stable generated chain of partial interpretations. This semantics can be viewed as a Tarski style model theory extending the *answer set semantics* of an extended logic program (ELP)[17] (in the case that all predicates are coherent). Indeed, according to our definitions of truth and falsity, the satisfaction and truth tables of the answer set semantics coincide with these for the coherent predicates of partial logic.

3 Extending RDF graphs with negative information

In this section, we extend RDF graphs to ERDF graphs, by adding strong negation. Moreover, we extend RDF graphs to ERDF formulas, which are built from positive ERDF triples, the connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$, and the quantifiers $\forall$, $\exists$.

According to RDF concepts [24, 18], *URI references* are used as globally unique names for web resources. An RDF URI reference is a Unicode string that represents an absolute URI (with optional fragment identifier). It may be represented as a *qualified name*, that is a colon-separated two-part string consisting of a *namespace prefix* (an abbreviated name for a namespace URI) and a local name. For example, given the namespace prefix "ex" defined to stand for the namespace URI "http://www.example.org/", the qualified name "ex:Riesling", which stands for "http://www.example.org/Reisling", is a URI reference.

A plain literal is a string "s", where s is a sequence of Unicode characters, or a pair of a string "s" and a language tag t, denoted by "s"@t. A typed literal

[7] The body of a general derivation rule is built using all connectives and quantifiers, whereas the head of the rule is built using the connectives $\neg$, $\wedge$, $\vee$.

is a pair of a string "s" and a datatype URI reference d, denoted by "s"^^d. For example, "27"^^$xsd{:}integer$ is a typed literal.

A (Web) *vocabulary* V is a set of URI references and/or literals (plain or typed). We denote the set of all URI references by URI, the set of all plain literals by $\mathcal{PL}$, the set of all typed literals by $\mathcal{TL}$, and the set of all literals by $\mathcal{LIT}$. It holds: $URI \cap \mathcal{LIT} = \emptyset$.

In our formalization, we consider a set Var of variable symbols, such that the sets Var, URI, $\mathcal{LIT}$ are pairwise disjoint. In the main text, variable symbols are explicitly indicated, while in our examples, variable symbols are prefixed by ?.

An RDF triple [24, 18] is a triple $s\ p\ o.$, where $s \in URI \cup Var$, $p \in URI$, and $o \in URI \cup \mathcal{LIT} \cup Var$, expressing that the subject s is related with the object o through the property p. An RDF graph is a set of RDF triples. The variable symbols appearing in an RDF graph are called *blank nodes*, and are, intuitively, *existentially quantified variables*. In this paper, we denote an RDF triple $s\ p\ o.$ also by $p(s, o)$. Below we extend the notion of RDF triple to allow for both positive and negative information.

Definition 1 (ERDF triple). Let V be a vocabulary. A *positive ERDF triple* over V (also called *ERDF sentence atom*) is an expression of the form $p(s, o)$, where $s, o \in V \cup Var$ are called *subject* and *object*, respectively, and $p \in V \cap URI$ is called *predicate* or *property*.
A *negative ERDF triple* over V is the strong negation $\neg p(s, o)$ of a positive ERDF triple $p(s, o)$ over V.
An *ERDF triple* over V (also called *ERDF sentence literal*) is a positive or negative ERDF triple over V. $\square$

We can also use the RDF-triple-like notation

$$s \quad -p \quad o\ .$$

for writing a negative ERDF triple and, as an option, use the + sign as a predicate prefix for marking positive triples, like in the following example:

ex:Gerd −ex:likes ex:CabernetSauvignon .
ex:Anastasia +ex:likes ex:CabernetSauvignon .
ex:Gerd +ex:likes ex:Riesling .
ex:Carlos −ex:likes ex:Riesling .

For example, $ex{:}likes(ex{:}Gerd, ex{:}Riesling)$ is a positive ERDF triple, expressing that *Gerd* likes *Riesling*, and $\neg ex{:}likes(ex{:}Carlos, ex{:}Riesling)$ is a negative ERDF triple, expressing that *Carlos* dislikes *Riesling*. Note that an RDF triple is a positive ERDF triple with the constraint that the subject of the triple is not a literal. For example, $ex{:}nameOf(\text{"}Grigoris\text{"}, ex{:}Grigoris)$ is a valid ERDF triple but not a valid RDF triple. Our choice of allowing literals appearing in the subject position is based on our intuition that this case can naturally appear in knowledge representation (as in the previous example). Moreover, note that a variable in the object position of an ERDF triple in the body of a rule, can appear in the subject position of the ERDF triple in the head of the rule. Since variables can be instantiated by a literal, a literal can naturally appear in the subject position of the derived ERDF triple.

Definition 2 (ERDF formula). Let V be a vocabulary. We consider the logical factors $\{\sim, \neg, \wedge, \vee, \supset, \exists, \forall\}$, where $\neg$, $\sim$, and $\supset$ are called *strong negation*, *weak negation*, and *material implication* respectively. We denote by $L(V)$ the smallest set that contains the positive ERDF triples over V and is closed with respect to the following conditions: if $F, G \in L(V)$ then $\{\sim F,\ \neg F,\ F \wedge G,\ F \vee G,\ F \supset G,\ \exists x F,\ \forall x F\} \subseteq L(V)$, where $x \in \mathit{Var}$. An *ERDF formula* over V is an element of $L(V)$. We denote the set of variables appearing in F by $\mathit{Var}(F)$, and the set of free variables[8] appearing in F by $F\mathit{Var}(F)$. Moreover, we denote set of URI references and literals appearing in F by V_F. $\square$

For example, let $F = \forall ?x\ \exists ?y\ (\mathit{rdf}{:}\mathit{type}(?x, \mathit{ex}{:}\mathit{Person}) \supset \mathit{ex}{:}\mathit{hasFather}(?x, ?y)) \wedge \mathit{rdf}{:}\mathit{type}(?z, \mathit{ex}{:}\mathit{Person})$. Then, F is an ERDF formula over the vocabulary $V = \{\mathit{rdf}{:}\mathit{type}, \mathit{ex}{:}\mathit{Person}, \mathit{ex}{:}\mathit{hasFather}\}$ with $\mathit{Var}(F) = \{?x, ?y, ?z\}$ and $F\mathit{Var}(F) = \{?z\}$.

We will denote the sublanguages of $L(V)$ formed by means of a subset S of the logical factors, by $L(V|S)$. For example, $L(V|\{\neg\})$ denotes the set of (positive and negative) ERDF triples over V.

Definition 3 (ERDF graph). An *ERDF graph* G is a set of ERDF triples over some vocabulary V. We denote the variables appearing in G by $\mathit{Var}(G)$, and the set of URI references and literals appearing in G by V_G. $\square$

Intuitively, an ERDF graph G represents an existentially quantified conjunction of *ERDF* triples. Specifically, let $G = \{t_1, ..., t_n\}$ be an *ERDF* graph, and let $\mathit{Var}(G) = \{x_1, ...x_k\}$. Then, G represents the formula $\exists x_1, ...x_k\ t_1 \wedge ... \wedge t_n$. Following the RDF terminology [24], the variables of an ERDF graph are called *blank nodes* and intuitively denote anonymous web resources.

For example, consider the ERDF graph $G = \{\mathit{rdf}{:}\mathit{type}(?x, \mathit{ex}{:}\mathit{EuropeanCountry}), \neg \mathit{rdf}{:}\mathit{type}(?x, \mathit{ex}{:}\mathit{EUmember})\}$. Intuitively, G denotes the ERDF formula $\exists ?x\ (\mathit{rdf}{:}\mathit{type}(?x, \mathit{ex}{:}\mathit{EuropeanCountry}) \wedge \neg \mathit{rdf}{:}\mathit{type}(?x, \mathit{ex}{:}\mathit{EUmember}))$, expressing that there is a European country which is not a European Union member.

Note that as an RDF graph is a set of RDF triples [24, 18], an RDF graph is also an ERDF graph.

4 ERDF Interpretations

In this section, we extend RDF(S) semantics by allowing for partial properties and classes. In particular, we define ERDF interpretations and satisfaction of an ERDF formula.

Below we define a partial interpretation as an extension of a simple interpretation [18], where each property is associated not only with a truth extension but also with a falsity extension allowing for partial properties.

Definition 4 (Partial interpretation). A *partial interpretation* I of a vocabulary V consists of:

– A non-empty set of resources Res_I, called the *domain* or *universe* of I.

[8] Without loss of generality, we assume that a variable cannot have both free and bound occurrences in F, and more than one bound occurrence.

– A set of properties $Prop_I$.
– A vocabulary interpretation mapping I_V[9]: $V \cap URI \to Res_I \cup Prop_I$.
– A property-truth extension mapping $PT_I : Prop_I \to \mathcal{P}(Res_I \times Res_I)$.
– A property-falsity extension mapping $PF_I : Prop_I \to \mathcal{P}(Res_I \times Res_I)$.
– A mapping $IL_I : V \cap \mathcal{TL} \to Res_I$.
– A set of literal values $LV_I \subseteq Res_I$, which contains $V \cap \mathcal{PL}$.

We define the mapping: $I : V \to Res_I \cup Prop_I$, called *denotation*, such that:

– $I(x) = I_V(x), \ \forall x \in V \cap URI$.
– $I(x) = x, \ \forall\, x \in V \cap \mathcal{PL}$.
– $I(x) = IL_I(x), \ \forall\, x \in V \cap \mathcal{TL}$. □

Definition 5 (Satisfaction of an ERDF formula w.r.t. a partial interpretation and a valuation). Let F, G be $ERDF$ formulas and let I be a partial interpretation of a vocabulary V. Let v be a mapping $v : Var(F) \to Res_I$ (called *valuation*). If $x \in Var(F)$, we define $[I+v](x) = v(x)$. If $x \in V$, we define $[I+v](x) = I(x)$.

– If $F = p(s,o)$ then $I, v \models F$ iff $p \in V \cap URI$, $s, o \in V \cup Var$, $I(p) \in Prop_I$, and $\langle [I+v](s), [I+v](o) \rangle \in PT_I(I(p))$.
– If $F = \neg p(s,o)$ then $I, v \models F$ iff $p \in V \cap URI$, $s, o \in V \cup Var$, $I(p) \in Prop_I$, and $\langle [I+v](s), [I+v](o) \rangle \in PF_I(I(p))$.
– If $F = {\sim}G$ then $I, v \models F$ iff $V_G \subseteq V$ and $I, v \not\models G$.
– If $F = F_1 \wedge F_2$ then $I, v \models F$ iff $I, v \models F_1$ and $I, v \models F_2$.
– If $F = F_1 \vee F_2$ then $I, v \models F$ iff $I, v \models F_1$ or $I, v \models F_2$.
– If $F = F_1 \supset F_2$ then $I, v \models F$ iff $I, v \models {\sim}F_1 \vee F_2$.
– If $F = \exists x\, G$ then $I, v \models F$ iff there exists mapping $u : Var(G) \to Res_I$ such that $u(y) = v(y)$, $\forall y \in Var(G) - \{x\}$, and $I, u \models G$.
– If $F = \forall x\, G$ then $I, v \models F$ iff for all mappings $u : Var(G) \to Res_I$ such that $u(y) = v(y)$, $\forall y \in Var(G) - \{x\}$, it holds $I, u \models G$.
– All other cases of $ERDF$ formulas are treated by the following DeMorgan-style rewrite rules expressing the falsification of compound ERDF formulas:
$\neg(F \wedge G) \to \neg F \vee \neg G$, $\neg(F \vee G) \to \neg F \wedge \neg G$, $\neg\neg F \to F$, $\neg \sim F \to F$,
$\neg \exists x\, F \to \forall x\, \neg F$, $\neg \forall x\, F \to \exists x\, \neg F$, $\neg(F \supset G) \to F \wedge \neg G$. □

Definition 6 (Satisfaction of an ERDF formula w.r.t. a partial interpretation). Let F be an $ERDF$ formula and let I be a partial interpretation of a vocabulary V. We say that I *satisfies* F, denoted by $I \models F$, iff for every mapping $v : Var(F) \to Res_I$, it holds $I, v \models F$. □

Similarly to first-order logic, the following proposition holds.

Proposition 1. Let F be an ERDF formula and let I be a partial interpretation of a vocabulary V. Let u, u' be mappings $u, u' : Var(F) \to Res_I$ such that $u(x) = u'(x), \ \forall x \in FVar(F)$. It holds: $I, u \models F$ iff $I, u' \models F$.

Note that as an ERDF graph represents an existentially quantified conjunction of ERDF triples (that is, an ERDF formula), Definition 6 applies also to ERDF graphs. Specifically, let G be an ERDF graph representing the formula $F = \exists x_1, ...x_k\ t_1 \wedge ... \wedge t_n$. We will show that a partial interpretation I *satisfies* the ERDF graph G ($I \models G$) iff $I \models F$.

The specific definition of ERDF graph satisfaction is given below, extending satisfaction of an RDF graph [18] (see also Appendix A).

[9] In the symbol I_V, V stands for *Vocabulary*.

Definition 7 (Satisfaction of an ERDF graph w.r.t. a partial interpretation). Let G be an $ERDF$ graph and let I be a partial interpretation of a vocabulary V. Let v be a mapping $v : Var(G) \rightarrow Res_I$. Then,

- $I, v \models G$ iff $\forall t \in G, \quad I, v \models t$.
- I $satisfies$ the ERDF graph G, denoted by $I \models G$, iff there exists a mapping $v : Var(G) \rightarrow Res_I$ such that $I, v \models G$. $\Box$

The following proposition is proved based on Proposition 1.

Proposition 2. Let $G = \{t_1, ..., t_n\}$ be an $ERDF$ graph and let $Var(G) = \{x_1, ..., x_k\}$. Let F be the ERDF formula $\exists x_1, ...x_k \ t_1 \wedge ... \wedge t_n$.
It holds: $I \models G$ iff $I \models F$.

V_{RDF}	V_{RDFS}
rdf:$type$	$rdfs$:$domain$
rdf:$Property$	$rdfs$:$range$
rdf:$XMLLiteral$	$rdfs$:$Resource$
rdf:nil	$rdfs$:$Literal$
rdf:$List$	$rdfs$:$Datatype$
rdf:$Statement$	$rdfs$:$Class$
rdf:$subject$	$rdfs$:$subClassOf$
rdf:$predicate$	$rdfs$:$subPropertyOf$
rdf:$object$	$rdfs$:$member$
rdf:$first$	$rdfs$:$Container$
rdf:$rest$	$rdfs$:$ContainerMembershipProperty$
rdf:Seq	$rdfs$:$comment$
rdf:Bag	$rdfs$:$seeAlso$
rdf:Alt	$rdfs$:$isDefinedBy$
rdf:$\_i$, $\forall i \in \{1, 2, ...\}$	$rdfs$:$label$
rdf:$value$	

Table 1. The vocabulary of RDF and RDFS

$$
\begin{array}{l}
rdf{:}type(rdf{:}type, rdf{:}Property) \\
rdf{:}type(rdf{:}subject, rdf{:}Property) \\
rdf{:}type(rdf{:}predicate, rdf{:}Property) \\
rdf{:}type(rdf{:}object, rdf{:}Property) \\
rdf{:}type(rdf{:}first, rdf{:}Property) \\
rdf{:}type(rdf{:}rest, rdf{:}Property) \\
rdf{:}type(rdf{:}value, rdf{:}Property) \\
rdf{:}type(rdf{:}\_i, rdf{:}Property), \quad \forall i \in \{1, 2, ...\} \\
rdf{:}type(rdf{:}nil, rdf{:}List)
\end{array}
$$

Table 2. The RDF axiomatic triple

The vocabulary of RDF, $\mathcal{V}_{RDF}$, is a set of *URI* references in the rdf: namespace [18], as shown in Table 1. The vocabulary of RDFS, $\mathcal{V}_{RDFS}$, is a set of *URI* references in the *rdfs*: namespace [18], as shown in Table 1.

The *vocabulary of ERDF*, $\mathcal{V}_{ERDF}$, is a set of *URI* references in the *erdf*: namespace. Specifically, the set of ERDF predefined classes is $\mathcal{C}_{ERDF} = \{erdf{:}TotalClass,\ erdf{:}TotalProperty\}$. We define $\mathcal{V}_{ERDF} = \mathcal{C}_{ERDF}$. Intuitively, instances of the metaclass $erdf{:}TotalClass$ are classes c that satisfy totalness, meaning that each resource belongs to the truth or falsity extension of c. Similarly, instances of the metaclass $erdf{:}TotalProperty$ are properties p that satisfy totalness, meaning that each pair of resources belongs to the truth or falsity extension of p.

We are now ready to define an ERDF interpretation over a vocabulary V as an extension of an RDFS interpretation [18] (see also Appendix A), where each property and class is associated not only with a truth extension but also with a falsity extension, allowing for both partial properties and partial classes. Additionally, an ERDF interpretation gives special semantics to terms from the ERDF vocabulary.

Definition 8 (ERDF interpretation). An *ERDF interpretation* I of a vocabulary V is a partial interpretation of $V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$, extended by the new ontological categories $Cls_I \subseteq Res_I$ for classes, $TCls_I \subseteq Cls_I$ for total classes, and $TProp_I \subseteq Prop_I$ for total properties, as well as the class-truth extension mapping $CT_I : Cls_I \to \mathcal{P}(Res_I)$, and the class-falsity extension mapping $CF_I : Cls_I \to \mathcal{P}(Res_I)$, such that:

1. $x \in CT_I(y)$ iff $\langle x, y\rangle \in PT_I(I(rdf{:}type))$, and
   $x \in CF_I(y)$ iff $\langle x, y\rangle \in PF_I(I(rdf{:}type))$.
2. The ontological categories are defined as follows:
   $Prop_I = CT_I(I(rdf{:}Property))$ $\quad Cls_I = CT_I(I(rdfs{:}Class))$
   $Res_I = CT_I(I(rdfs{:}Resource))$ $\quad LV_I = CT_I(I(rdfs{:}Literal))$
   $TCls_I = CT_I(I(erdf{:}TotalClass))$ $TProp_I = CT_I(I(erdf{:}TotalProperty))$.
3. if $\langle x, y\rangle \in PT_I(I(rdfs{:}domain))$ and $\langle z, w\rangle \in PT_I(x)$ then $z \in CT_I(y)$.
4. If $\langle x, y\rangle \in PT_I(I(rdfs{:}range))$ and $\langle z, w\rangle \in PT_I(x)$ then $w \in CT_I(y)$.
5. If $x \in Cls_I$ then $\langle x, I(rdfs{:}Resource)\rangle \in PT_I(I(rdfs{:}subclassOf))$.
6. If $\langle x, y\rangle \in PT_I(I(rdfs{:}subClassOf))$ then $x, y \in Cls_I$, $CT_I(x) \subseteq CT_I(y)$, and $CF_I(y) \subseteq CF_I(x)$.
7. $PT_I(I(rdfs{:}subClassOf))$ is a reflexive and transitive relation on Cls_I.
8. If $\langle x, y\rangle \in PT_I(I(rdfs{:}subPropertyOf))$ then $x, y \in Prop_I$, $PT_I(x) \subseteq PT_I(y)$, and $PF_I(y) \subseteq PF_I(x)$.
9. $PT_I(I(rdfs{:}subPropertyOf))$ is a reflexive and transitive relation on $Prop_I$.
10. If $x \in CT_I(I(rdfs{:}Datatype))$ then $\langle x, I(rdfs{:}Literal)\rangle \in PT_I(I(rdfs{:}subClassOf))$.
11. If $x \in CT_I(I(rdfs{:}ContainerMembershipProperty))$ then $\langle x, I(rdfs{:}member)\rangle \in PT_I(I(rdfs{:}subPropertyOf))$.
12. If $x \in TCls_I$ then $CT_I(x) \cup CF_I(x) = Res_I$.
13. If $x \in TProp_I$ then $PT_I(x) \cup PF_I(x) = Res_I \times Res_I$.
14. If "s"^^$rdf{:}XMLLiteral \in V$ and s is a well-typed XML literal string, then $IL_I($"s"^^$rdf{:}XMLLiteral)$ is the XML value of s, and
    $IL_I($"s"^^$rdf{:}XMLLiteral) \in CT_I(I(rdf{:}XMLLiteral))$.
15. If "s"^^$rdf{:}XMLLiteral \in V$ and s is an ill-typed XML literal string then
    $IL_I($"s"^^$rdf{:}XMLLiteral) \in Res_I - LV_I$, and
    $IL_I($"s"^^$rdf{:}XMLLiteral) \in CF_I(I(rdfs{:}Literal))$.
16. I satisfies the RDF and $RDFS$ axiomatic triples [18], shown in Table 2 and Table 3, respectively.
17. I satisfies the following triples, called *ERDF axiomatic triples*:
    $rdfs{:}subClassOf(erdf{:}TotalClass, rdfs{:}Class)$.
    $rdfs{:}subClassOf(erdf{:}TotalProperty, rdf{:}Property)$.

$rdfs{:}domain(rdf{:}type, rdfs{:}Resource)$
$rdfs{:}domain(rdfs{:}domain, rdf{:}Property)$
$rdfs{:}domain(rdfs{:}range, rdf{:}Property)$
$rdfs{:}domain(rdfs{:}subpropertyOf, rdf{:}Property)$
$rdfs{:}domain(rdfs{:}subClassOf, rdfs{:}Class)$
$rdfs{:}domain(rdf{:}subject, rdf{:}Statement)$
$rdfs{:}domain(rdf{:}predicate, rdf{:}Statement)$
$rdfs{:}domain(rdf{:}object, rdf{:}Statement)$
$rdfs{:}domain(rdfs{:}member, rdfs{:}Resource)$
$rdfs{:}domain(rdf{:}first, rdf{:}List)$
$rdfs{:}domain(rdf{:}rest, rdf{:}List)$
$rdfs{:}domain(rdfs{:}seeAlso, rdfs{:}Resource)$
$rdfs{:}domain(rdfs{:}isDefinedBy, rdfs{:}Resource)$
$rdfs{:}domain(rdfs{:}comment, rdfs{:}Resource)$
$rdfs{:}domain(rdfs{:}label, rdfs{:}Resource)$
$rdfs{:}domain(rdfs{:}value, rdfs{:}Resource)$
$rdfs{:}range(rdf{:}type, rdfs{:}Class)$
$rdfs{:}range(rdfs{:}domain, rdfs{:}Class)$
$rdfs{:}range(rdfs{:}range, rdfs{:}Class)$
$rdfs{:}range(rdfs{:}subPropertyOf, rdf{:}Property)$
$rdfs{:}range(rdfs{:}subClassOf, rdfs{:}Class)$
$rdfs{:}range(rdf{:}subject, rdfs{:}Resource)$
$rdfs{:}range(rdf{:}predicate, rdfs{:}Resource)$
$rdfs{:}range(rdf{:}object, rdfs{:}Resource)$
$rdfs{:}range(rdfs{:}member, rdfs{:}Resource)$
$rdfs{:}range(rdf{:}first, rdfs{:}Resource)$
$rdfs{:}range(rdf{:}rest, rdf{:}List)$
$rdfs{:}range(rdfs{:}seeAlso, rdfs{:}Resource)$
$rdfs{:}range(rdfs{:}isDefinedBy, rdfs{:}Resource)$
$rdfs{:}range(rdfs{:}comment, rdfs{:}Literal)$
$rdfs{:}range(rdfs{:}label, rdfs{:}Literal)$
$rdfs{:}range(rdf{:}value, rdfs{:}Resource)$
$rdfs{:}subClassOf(rdf{:}Alt, rdfs{:}Container)$
$rdfs{:}subClassOf(rdf{:}Bag, rdfs{:}Container)$
$rdfs{:}subClassOf(rdf{:}Seq, rdfs{:}Container)$
$rdfs{:}subClassOf(rdfs{:}ContainerMembershipProperty, rdf{:}Property)$
$rdfs{:}subPropertyOf(rdfs{:}isDefinedBy, rdfs{:}seeAlso)$
$rdf{:}type(rdf{:}XMLLiteral, rdfs{:}Datatype)$
$rdfs{:}subClassOf(rdf{:}XMLLiteral, rdfs{:}Literal)$
$rdfs{:}subClassOf(rdfs{:}Datatype, rdfs{:}Class)$
$rdf{:}type(rdf{:}\_i, rdfs{:}ContainerMembershipProperty), \quad \forall i \in \{1, 2, ...\}$
$rdfs{:}domain(rdf{:}\_i, rdfs{:}Resource), \quad \forall i \in \{1, 2, ...\}$
$rdfs{:}range(rdf{:}\_i, rdfs{:}Resource), \quad \forall i \in \{1, 2, ...\}$

Table 3. The RDFS axiomatic triples

Note that the semantic conditions of ERDF interpretations may impose constraints to both the truth and falsity extensions of properties and classes. For example, consider semantic condition 6 of Definition 8 and assume that $\langle x, y \rangle \in PT_I(I(rdfs{:}subClassOf))$. Then, I should satisfy not only $CT_I(x) \subseteq CT_I(y)$, but also $CF_I(y) \subseteq CF_I(x)$. Similar is the case for semantic conditions 8, 12, 13, 14, and 17.

Definition 9 (Coherent ERDF interpretation). An ERDF interpretation I of a vocabulary V is *coherent* iff for all $x \in Prop_I$, $PT_I(x) \cap PF_I(x) = \emptyset$. $\square$

Coherent ERDF interpretations enforce the constraint that a pair of resources cannot belong to both the truth and falsity extensions of a property. Intuitively, this means that an ERDF triple cannot be both true and false. Since *rdf:type* is a property, this constraint also implies that a resource cannot belong to both the truth and falsity extensions of a class.

Proposition 3. Let I be a coherent ERDF interpretation of a vocabulary V. It holds: $\forall x \in Cls_I, \quad CT_I(x) \cap CF_I(x) = \emptyset$.

Thus, all properties and classes of coherent ERDF interpretations are coherent.

In the rest of the document, we consider only coherent ERDF interpretations. This means that referring to an "ERDF interpretation", we implicitly mean a "coherent" one.

According to RDFS semantics [18], the only source of RDFS-inconsistency is the appearance of an ill-typed XML literal l in the RDF graph, in combination with the derivation of the RDF triple "*x rdf:type rdfs:Literal.*" by the RDF and RDFS entailment rules, where x is a blank node allocated to l (for details, see [18]) Such a triple is called *XML clash*. An ERDF graph can be ERDF-inconsistent[10], not only due to the appearance of an ill-typed XML literal in the ERDF graph (in combination with the semantic condition 15), but also due to the additional semantic condition for coherent ERDF interpretations.

For example, let $p, q, s, o \in URI$ and let $G = \{p(s, o),\ rdfs{:}subPropertyOf(p, q),\ \neg q(s, o)\}$. Then, G is ERDF-inconsistent, since there is no (coherent) ERDF interpretation that satisfies G.

The following proposition shows that for total properties and total classes of (coherent) ERDF interpretations, weak negation and strong negation coincide (boolean truth values).

Proposition 4. Let I be an ERDF interpretation of a vocabulary V and let $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Then,

1. For all $p, s, o \in V'$, such that $I(p) \in TProp_I$, it holds:
   $I \models \sim p(s, o)$ iff $I \models \neg p(s, o)$ (equivalently, $I \models p(s, o) \vee \neg p(s, o)$).
2. For all $x, c \in V'$ such that $I(c) \in TCls_I$, it holds:
   $I \models \sim rdf{:}type(x, c)$ iff $I \models \neg rdf{:}type(x, c)$
   (equivalently, $I \models rdf{:}type(x, c) \vee \neg rdf{:}type(x, c)$).

[10] Meaning that there is no (coherent) ERDF interpretation that satisfies the ERDF graph.

Definition 10 (Classical ERDF interpretation). A (coherent) ERDF interpretation I of a vocabulary V is *classical* iff for all $x \in Prop_I$, $PT_I(x) \cup PF_I(x) = Res_I \times Res_I$. $\Box$

A classical ERDF interpretation is close to an interpretation of classical logic, since for every formula F, weak and strong negation coincide.

Proposition 5. Let I be an ERDF interpretation of a vocabulary V and let $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Then,

1. If $TProp_I = Prop_I$ then I is a classical ERDF interpretation.
2. If I is a classical ERDF interpretation and F is an ERDF formula over V' such that $I(p) \in Prop_I$, for every property p in F, then it holds: $I \models \sim F$ iff $I \models \neg F$ (equivalently, $I \models F \vee \neg F$).

Below we define ERDF entailment between two ERDF formulas or ERDF graphs.

Definition 11 (ERDF Entailment). Let F, F' be ERDF formulas or ERDF graphs. We say that F *ERDF-entails* F' ($F \models^{ERDF} F'$) iff for every ERDF interpretation I, if $I \models F$ then $I \models F'$. $\Box$

For example, let $F = \forall ?x\ \exists ?y\ (rdf{:}type(?x, ex{:}Person) \supset ex{:}hasFather(?x, ?y))$ $\wedge\ rdf{:}type(ex{:}John, ex{:}Person)$, and let $F' = \exists ?y\ ex{:}hasFather(ex{:}John, ?y)$ $\wedge\ rdf{:}type(ex{:}hasFather, rdf{:}Property)$. Then $F \models^{ERDF} F'$.

The following proposition shows that an RDF graph is RDFS satisfiable iff it is ERDF satisfiable. Thus, an RDF graph can be ERDF-inconsistent only due to an XML clash.

Proposition 6. Let G be an RDF graph such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$. Then, there is an RDFS interpretation that satisfies G iff there is an ERDF interpretation that satisfies G.

The following proposition shows that ERDF entailment extends RDFS entailment [18] (see also Appendix A) from RDF graphs to ERDF formulas. In other words, ERDF entailment is upward compatible with RDFS entailment.

Proposition 7. Let G, G' be RDF graphs such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$ and $V_{G'} \cap \mathcal{V}_{ERDF} = \emptyset$. Then, $G \models^{RDFS} G'$ iff $G \models^{ERDF} G'$.

5 ERDF Ontologies

In this section, we define an ERDF ontology as a pair of an ERDF graph G and a set P of ERDF rules. ERDF rules should be considered as derivation rules that allow us to infer more ontological information based on the declarations in G. Moreover, we define the Herbrand interpretations and the Herbrand models of an ERDF ontology.

Definition 12 (ERDF rule, ERDF program). An *ERDF rule* r over a vocabulary V is an expression of the form: $G \leftarrow F$, where $F \in L(V) \cup \{true\}$ is called *condition* and $G \in L(V|\{\neg\})$ is called *conclusion*. We assume that no bound variable in F appears free in G. We denote the set of variables and the

set of free variables of r by $Var(r)$ and $FVar(r)$[11], respectively. Additionally, we write $Cond(r) = F$ and $Concl(r) = G$.
An *ERDF program* P is a set of ERDF rules over some vocabulary V. We denote the set of URI references and literals appearing in P by V_P. □

For example, the following derivation rule r is an ERDF rule:

$$ex{:}allRelated(?P, ?R) \leftarrow \forall ?p \;\; rdf{:}type(?p, ?P) \supset \exists ?r \;\; rdf{:}type(?r, ?R) \wedge ex{:}related(?p, ?r),$$

indicating that between two class P, R it holds $ex{:}allRelated(P, R)$ if for all instances p of the class P, there is an instance r of the class R such that it holds $ex{:}related(p, r)$. It holds $Var(r) = \{?P, ?R, ?p, ?r\}$ and $FVar(r) = \{?P, ?R\}$.

When $Cond(r) = true$ and $Var(r) = \{\}$, rule r is also called ERDF fact. We assume that for every partial interpretation I, it holds $I \models true$.

Intuitively, an ERDF ontology is the combination of (i) an ERDF graph G containing (implicitly existentially quantified) positive and negative information, and (ii) an ERDF program P containing derivation rules (whose free variables are implicitly universally quantified).

Definition 13 (ERDF ontology). An *ERDF ontology* (or *knowledge base*) is a pair $O = \langle G, P \rangle$, where G is an ERDF graph and P is an ERDF program. □

The following definition defines the models of an ERDF ontology.

Definition 14 (Satisfaction of an ERDF rule and an ERDF ontology). Let I be an ERDF interpretation of a vocabulary V.

- We say that I *satisfies* an ERDF rule r, denoted by $I \models r$, iff: For all mappings $v : Var(r) \rightarrow Res_I$ such that $I, v \models Cond(r)$, it holds $I, v \models Concl(r)$.
- We say that I *satisfies* an ERDF ontology $O = \langle G, P \rangle$ (also, I is a *model* of O), denoted by $I \models O$, iff $I \models G$ and $I \models r$, $\forall\, r \in P$. □

In this paper, existentially quantified variables in ERDF graphs are handled by
skolemization, a syntactic transformation commonly used in automatic inference systems for removing existentially quantified variables.

Definition 15 (Skolemization of an ERDF graph). Let G be an ERDF graph. The *skolemization function* of G is an 1:1 mapping $sk_G : Var(G) \rightarrow URI$, where for each $x \in Var(G)$, $sk_G(x)$ is an artificial URI, denoted by $G{:}x$. The set $sk_G(Var(G))$ is called the *Skolem vocabulary* of G.
The *skolemization* of G, denoted by $sk(G)$, is the ground ERDF graph derived from G after replacing each variable $x \in Var(G)$ by $sk_G(x)$. □

Intuitively, the Skolem vocabulary of G (that is, $sk_G(Var(G))$) contains artificial URIs giving "arbitrary" names to the anonymous entities whose existence was asserted by the use of blank nodes in G.

For example, let $G = \{rdf{:}type(?x, ex{:}EuropeanCountry), \neg rdf{:}type(?x, ex{:}EUmember)\}$. Then $sk(G) = \{rdf{:}type(sk_G(?x), ex{:}EuropeanCountry), \neg rdf{:}type(sk_G(?x), ex{:}EUmember)\}$.

As the following proposition shows, skolemization preserves satisfiability.

[11] $FVar(r) = FVar(F) \cup FVar(G)$.

Proposition 8. Let G be an ERDF graph. There is an ERDF interpretation that satisfies G iff there is an ERDF interpretation that satisfies $sk(G)$.

Below we show that if an ERDF interpretation satisfies the skolemization of an ERDF graph then it also satisfies the original graph.

Proposition 9. Let G be an ERDF graph and let I be an ERDF interpretation. Then, $I \models sk(G)$ implies $I \models G$.

The following proposition expresses that the skolemization of an ERDF graph has the same entailments as the original graph, provided that these do not contain URIs from the skolemization vocabulary.

Proposition 10. Let G be an ERDF graph and F be an ERDF formula such that $V_F \cap sk_G(Var(G)) = \emptyset$. It holds: $G \models^{ERDF} F$ iff $sk(G) \models^{ERDF} F$.

Below we define the vocabulary of an ERDF ontology O.

Definition 16 (Vocabulary an ERDF ontology). Let $O = \langle G, P \rangle$ be an ERDF ontology. The *vocabulary* of O is defined as $V_O = V_{sk(G)} \cup V_P \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. $\square$

Note that the vocabulary of an ontology $O = \langle G, P \rangle$ contains the skolemization vocabulary of G.

Let $O = \langle G, P \rangle$ be an ERDF ontology. We denote by Res^H_O the union of V_O and the set of XML values of the well-typed XML literals in V_O minus the well-typed XML literals.

The following definition defines the Herbrand interpretations and the Herbrand models of an ERDF ontology.

Definition 17 (Herbrand interpretation, Herbrand model of an ERDF ontology). Let $O = \langle G, P \rangle$ be an ERDF ontology and let I be an ERDF interpretation of V_O. I is a *Herbrand interpretation* of O iff:

- $Res_I = Res^H_O$.
- $I_V(x) = x$, for all $x \in V_O \cap URI$.
- $IL_I(x) = x$, if x is a typed literal in V_O other than a well-typed XML literal, and $IL_I(x)$ is the XML value of x, if x is a well-typed XML literal in V_O.

We denote the set of Herbrand interpretations of O by $\mathcal{I}^H(O)$.
A Herbrand interpretation I of O is a *Herbrand model* of O iff $I \models \langle sk(G), P \rangle$.
We denote the set of Herbrand models of O by $\mathcal{M}^H(O)$. $\square$

Note that if I is a Herbrand interpretation of an ERDF ontology O then $I(x) = x$, for each $x \in V_O$ other than a well-typed XML literal.

Obviously, every Herbrand model of an ERDF ontology O is a model of O.

6 Minimal Herbrand Interpretations and Stable Models

In the previous section, we defined the Herbrand models of an ERDF ontology O. However, not all Herbrand models of O are desirable. In this section, we define the intended models of O, called *stable models* of O, based on minimal Herbrand interpretations. In particular, defining the stable models of O, only the minimal interpretations from a set of Herbrand interpretations that satisfy certain criteria are considered.

For example, let $p, s, o \in \mathit{URI}$, let $G = \{p(s, o)\}$, and let $O = \langle G, \emptyset \rangle$, Then, there is a Herbrand model I of O such that $I \models p(o, s)$, whereas we want $\sim p(o, s)$ to be satisfied by all intended models of O, as p is not a total property[12] and $p(o, s)$ cannot be derived from O (negation-as-failure).

To define the minimal Herbrand interpretations of an ERDF ontology O, we need to define a partial ordering on the Herbrand interpretations of O.

Definition 18 (Herbrand interpretation ordering). Let $O = \langle G, P \rangle$ be an ERDF ontology. Let $I, J \in \mathcal{I}^H(O)$. We say that J *extends* I, denoted by $I \leq J$ (or $J \geq I$), iff $Prop_I \subseteq Prop_J$, and for all $p \in Prop_I$, it holds $PT_I(p) \subseteq PT_J(p)$ and $PF_I(p) \subseteq PF_J(p)$. $\Box$

It is easy to verify that $\leq$ is indeed a partial ordering on $\mathcal{I}^H(O)$, as it is reflexive, transitive, and antisymmetric[13].

The intuition behind Definition 18 is that by extending a Herbrand interpretation, we extend both the truth and falsity extension for all properties, and thus (since *rdf:type* is a property), for all classes.

Proposition 11. Let $O = \langle G, P \rangle$ be an ERDF ontology and let $I, J \in \mathcal{I}^H(O)$ such that $I \leq J$. Then, $Cls_I \subseteq Cls_J$, and for all $c \in Cls_I$, it holds $CT_I(c) \subseteq CT_J(c)$ and $CF_I(c) \subseteq CF_J(c)$.

Definition 19 (Minimal Herbrand Interpretations). Let O be an ERDF ontology and let $\mathcal{I} \subseteq \mathcal{I}^H(O)$. We define $minimal(\mathcal{I}) = \{I \in \mathcal{I} \mid \ \nexists J \in \mathcal{I} : \ J \neq I$ and $J \leq I\}$. $\Box$

Let $I, J \in \mathcal{I}^H(O)$, we define $[I, J]_O = \{I' \in \mathcal{I}^H(O), \ I \leq I' \leq J\}$.

Additionally, we define the *minimal Herbrand models* of O, as:
$\mathcal{M}^{min}(O) = minimal(\mathcal{M}^H(O))$.

However minimal Herbrand models do not give the intended semantics to all ERDF rules. This is because ERDF rules are derivation and not implication rules. Derivation rules are often identified with implications. But, in general, these are two different concepts. While an implication is an expression of a logical formula language, a derivation rule is rather a meta-logical expression. There are logics, which do not have an implication connective, but which have a derivation rule concept. In standard logics (such as classical and intuitionistic logic), there is a close relationship between a derivation rule (also called "sequent") and the corresponding implicational formula: they both have the same models. For nonmonotonic rules (e.g. with negation-as-failure), this is no longer the case: the intended models of such a rule are, in general, not the same as the intended models of the corresponding implication. This is easy to see with help of an

[12] For total predicates, which are synonymous to classical predicates in this paper, the LEM applies. Thus, if p is a total property, then $p(o, s) \vee \neg p(o, s)$ should be satisfied by all intended models and, hence, $\sim p(o, s)$ is not satisfied.

[13] Indeed, let $I, J \in \mathcal{I}^H(O)$ s.t. $I \leq J$ and $J \leq I$. Then, I, J are ERDF interpretations of V_O such that $Res_I = Res_J$, $Prop_I = Prop_J$, $I_V = J_V$, $PT_I = PT_J$, $PF_I = PF_J$, $IL_I = IL_J$, $Cls_I = Cls_J$, $CT_I = CT_J$, $TProp_I = TProp_J$, and $TCls_I = TCls_J$. Thus, $I = J$.

example. Consider the rule $\sim q \rightarrow p$ whose model set, according to the stable model semantics [16, 17, 20, 19], is $\{\{p\}\}$, that is, it entails p. On the other hand, the model set of the corresponding implication $\sim q \supset p$ which is equivalent to the disjunction $p \vee q$, is $\{\{p\}, \{q\}, \{p, q\}\}$; consequently, it does not entail p.

Similarly, let $O = \langle \emptyset, P \rangle$, where $P = \{p(s, o) \leftarrow \sim q(s, o)\}$ and $p, q, s, o \in \mathit{URI}$. Not all minimal Herbrand models of O are intended. In particular, there is $I \in \mathcal{M}^{min}(O)$ such that $I \models q(s, o) \wedge \sim p(s, o)$, whereas we want $\sim q(s, o) \wedge p(s, o)$ to be satisfied by all intended models of O, as q is not a total property and $q(s, o)$ cannot be derived by any rule (negation-as-failure).

To define the intended (*stable*) models of an ERDF ontology, we need first to define grounding of ERDF rules.

Definition 20 (Grounding of an ERDF program). Let V be a vocabulary and r be an ERDF rule. We denote by $[r]_V$ the set of rules that result from r if we replace each variable $x \in \mathit{FVar}(r)$ by $v(x)$, for all mappings $v : \mathit{FVar}(r) \rightarrow V$. Let P be an ERDF program. We define $[P]_V = \bigcup_{r \in P} [r]_V$. $\square$

Below, we define the stable models of an ERDF ontology based on the coherent stable models of partial logic [19] (which, on extended logic programs, are equivalent [19] to Answer Sets of answer set semantics [17]).

Definition 21 (Stable model). Let $O = \langle G, P \rangle$ be an ERDF ontology and let $M \in \mathcal{I}^H(O)$. We say that M is a *stable model* of O iff there is a chain of Herbrand interpretations of O, $I_0 \leq ... \leq I_k$ such that $I_{k-1} = I_k = M$ and:

1. $I_0 \in \mathit{minimal}(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})$.
2. For $0 < \alpha \leq k$:
   $I_\alpha \in \mathit{minimal}\{I \in \mathcal{I}^H(O) \mid I \geq I_{\alpha-1}$ and $I \models \mathit{Concl}(r)$, for all $r \in P_{[I_{\alpha-1}, M]}\}$, where
   $P_{[I_{\alpha-1}, M]} = \{r \in [P]_{V_O} \mid I \models \mathit{Cond}(r),\ \forall I \in [I_{\alpha-1}, M]_O\}$.

The set of stable models of O is denoted by $\mathcal{M}^{st}(O)$. $\square$

Let us see an example. Consider an example namespace *ex*:, a class *ex*:*Paper* whose instances are papers submitted to a conference, a class *ex*:*Reviewer* whose instances are potential reviewers for the submitted papers, and a property $ex{:}conflict(R, P)$ indicating that there is a conflict of interest between reviewer R and paper P. Assume that we want to assign papers to reviewers based only on the following criteria: (i) a paper is assigned to at most one reviewer, (ii) a reviewer is assigned at most one paper, and (iii) no paper is assigned to a reviewer with whom there is conflict of interest. The assignment of a paper P to a reviewer R is indicated through the property $ex{:}assign(P, R)$. The ERDF triple $ex{:}allAssigned(ex{:}Paper, ex{:}Reviewer)$ indicates that each paper has been assigned to one reviewer. Ignoring for simplicity the example namespace *ex*:, the ERDF program P describing assignment of papers is the following (commas "," in the body of the rules indicate conjunction $\wedge$):

$$
\begin{array}{ll}
id(?x, ?x) & \leftarrow true. \\
\neg assign(?p, ?r) & \leftarrow rdf{:}type(?p, Paper), rdf{:}type(?p', Paper), assign(?p', ?r), \\
 & \quad \sim id(?p, ?p'). \\
\neg assign(?p, ?r) & \leftarrow rdf{:}type(?r, Reviewer), rdf{:}type(?r', Reviewer), assign(?p, ?r'), \\
 & \quad \sim id(?r, ?r'). \\
\neg assign(?p, ?r) & \leftarrow conflict(?r, ?p). \\
assign(?p, ?r) & \leftarrow rdf{:}type(?r, Reviewer), rdf{:}type(?p, Paper), \sim \neg assign(?p, ?r). \\
\multicolumn{2}{l}{allAssigned(Paper, Reviewer) \leftarrow \forall ?p(; rdf{:}type(?p, Paper) \supset} \\
\multicolumn{2}{r}{\exists ?r\ \ rdf{:}type(?r, Reviewer) \wedge assign(?p, ?r)).}
\end{array}
$$

Consider now the ERDF graph G, containing the factual information:

$G = \{$ *rdf:type*$(P1, Paper)$, *rdf:type*$(P2, Paper)$, *rdf:type*$(P3, Paper)$,
rdf:type$(R1, Reviewer)$, *rdf:type*$(R2, Reviewer)$, *rdf:type*$(R3, Reviewer)$,
$conflict(P1, R3)$, $conflict(P2, R2)$, $conflict(P3, R2)\}$.

Then, according to Definition 21, the ERDF ontology $O = \langle G, P \rangle$ has four stable models, denoted by M_1, ..., M_4, such that:

$M_1 \models assign(P1, R1) \wedge assign(P2, R3) \wedge \sim allAssigned(Paper, Reviewer)$,
$M_2 \models assign(P1, R1) \wedge assign(P3, R3) \wedge \sim allAssigned(Paper, Reviewer)$,
$M_3 \models assign(P1, R2) \wedge assign(P2, R1) \wedge assign(P3, R3) \wedge$
$allAssigned(Paper, Reviewer)$, and
$M_4 \models assign(P1, R2) \wedge assign(P2, R3) \wedge assign(P3, R1) \wedge$
$allAssigned(Paper, Reviewer)$.

The following proposition shows that a stable model of an ERDF ontology O is a Herbrand model of O.

Proposition 12. Let $O = \langle G, P \rangle$ be an ERDF ontology and let $M \in \mathcal{M}^{st}(O)$. It holds $M \in \mathcal{M}^H(O)$.

On the other hand, if all properties are total, a Herbrand model M of an ERDF ontology $O = \langle G, P \rangle$ is a stable model of O. This is because, in this case $M \in minimal(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})$ and $M \in minimal\{I \in \mathcal{I}^H(O) \mid I \geq M$ and $I \models Concl(r)$, for all $r \in P_{[M,M]}\}$.

Proposition 13. Let $O = \langle G, P \rangle$ be an ERDF ontology, such that *rdfs:subclass*(*rdf:Property*, *erdf:TotalProperty*) $\in G$. Then, $\mathcal{M}^{st}(O) = \mathcal{M}^H(O)$.

¿From Proposition 5, it follows that if *rdfs:subclass*(*rdf:Property*, *erdf:TotalProperty*) $\in G$ then each $M \in \mathcal{M}^H(O)$ is a classical ERDF interpretation. Therefore, the above proposition shows that classical (boolean) Herbrand model reasoning on ERDF ontologies is a special case of stable model reasoning.

Similarly to [16, 17, 20, 19], stable models do not preserve Herbrand model satisfiability. For example, let $O = \langle \emptyset, P \rangle$, where $P = \{p(s, o) \leftarrow \sim p(s, o)\}$, and $p, s, o \in URI$. Then, $\mathcal{M}^{st}(O) = \emptyset$, whereas there is a Herbrand model of O that satisfies $p(s, o)$.

Below we define stable model entailment on ERDF ontologies.

Definition 22. Stable model entailment
Let $O = \langle G, P \rangle$ be an ERDF ontology and let F be an ERDF formula or ERDF graph. We say that O *entails* F under the *(ERDF) stable model semantics*, denoted by $O \models^{st} F$ iff for all $M \in \mathcal{M}^{st}(O)$, $M \models F$. □

For example, let $O = \langle \emptyset, P \rangle$, where $P = \{p(s, o) \leftarrow \sim q(s, o)\}$ and $p, q, s, o \in URI$. Then, $O \models^{st} \sim q(s, o) \wedge p(s, o)$. Let $O = \langle G, P \rangle$, where $G = \{$*rdfs:subclass*(*rdf:Property*, *erdf:TotalProperty*)$\}$ and P is as in the previous example. Then, $O \models^{st} q(s, o) \vee p(s, o)$, but $O \not\models^{st} \sim q(s, o)$ and $O \not\models^{st} p(s, o)$. This is the desirable result, since q is a total property, and thus in contrast to the previous example, an open-world assumption is made for q. As another example, let $p, s, o \in URI$, let $G = \{p(s, o)\}$, and let $P = \{\neg p(?x, ?y) \leftarrow \sim p(?x, ?y)\}$. Then, $\langle G, P \rangle \models^{st} \sim p(o, s) \wedge \neg p(o, s)$ (note that P contains a CWA on p). Let

$G = \{rdf\text{:}type(p, erdf\text{:}TotalProperty),\ p(s,o)\}$ and let P be as in the previous example. Then, $\langle G, P\rangle \models^{st} \forall ?x\ \forall ?y\ \ (p(?x,?y) \vee \neg p(?x,?y))$ (see Proposition 4), but $\langle G, P\rangle \not\models^{st} \sim p(o,s)$ and $\langle G, P\rangle \not\models^{st} \neg p(o,s)$. Indeed, the CWA in P does not affect the semantics of p, since p is a total property.

Let us now see a more involved example[14]. Consider the following ERDF program P, specifying some rules for concluding that a country is not a member state of the European Union (EU).

$$\begin{array}{lll} (r_1) & \neg rdf\text{:}type(?x, EUMember) \leftarrow & rdf\text{:}type(?x, AmericanCountry). \\ (r_2) & \neg rdf\text{:}type(?x, EUMember) \leftarrow & rdf\text{:}type(?x, EuropeanCountry), \\ & & \sim rdf\text{:}type(?x, EUMember). \end{array}$$

A rather incomplete ERDF ontology $O = \langle G, P\rangle$ is obtained by including the following information in the ERDF graph G:

$$\begin{array}{ll} \neg rdf\text{:}type(Russia, EUMember). & rdf\text{:}type(Canada, AmericanCountry). \\ rdf\text{:}type(Austria, EUMember). & rdf\text{:}type(Italy, EuropeanCountry). \\ rdf\text{:}type(?x, EuropeanCountry). & \neg rdf\text{:}type(?x, EUMember). \end{array}$$

Using stable model entailment on O, it can be concluded that Austria is a member of EU, that Russia and Canada are not members of EU, and that it exists a European Country which is not a member of EU. However, it is also concluded that Italy is not a member of EU, which is a wrong statement. This is because G does not contain complete information of the European countries that are EU members (e.g., it does not contain $rdf\text{:}type(Italy, EUMember)$). Thus, incorrect information is obtained by the closed-world assumption expressed in rule r_2. In the case that $rdf\text{:}type(EUMember, erdf\text{:}TotalClass)$ is added to G (that is, an open-world assumption is made for the class $EUMember$) then $\sim rdf\text{:}type(Italy, EUMember)$ and thus, $\neg rdf\text{:}type(Italy, EUMember)$ are not longer entailed. This is because, there is a stable model of the extended O that satisfies $rdf\text{:}type(Italy, EUMember)$. Moreover, if complete information for all European countries that are members of EU is included in G then the stable model conclusions of O will also be correct (the closed-world assumption will be correctly applied). Note that, in this case G will include $rdf\text{:}type(Italy, EUMember)$.

Proposition 14. Let $O = \langle G, P\rangle$ be an ERDF ontology, and let F, F' be ERDF formulas. If $O \models^{st} F$ and $F \models^{ERDF} F'$ then $O \models^{st} F'$.

The following proposition, together with Proposition 10, shows that stable model entailment on ERDF ontologies is upward compatible with ERDF entailment on ERDF graphs.

Proposition 15. Let G, G' be ERDF graphs and F be an ERDF formula.
It holds:

1. If $\langle G, \emptyset\rangle \models^{st} G'$ then $sk(G) \models^{ERDF} G'$.
2. If $sk(G) \models^{ERDF} F$ then $\langle G, \emptyset\rangle \models^{st} F$.

Let $G = \{p(s,o)\}$, where $p, s, o \in URI$. Then $\langle G, \emptyset\rangle \models^{st} \sim p(o,s)$, whereas $sk(G) \not\models^{ERDF} \sim p(o,s)$. This shows that the first statement of Proposition 15 cannot be generalized from an ERDF graph G' to any ERDF formula F. Let

[14] For simplicity, the example namespace *ex*: is ignored.

G, G' be ERDF graphs. It follows from Propositions 10 and 15 that $G \models^{ERDF} G'$ iff $\langle G, \emptyset \rangle \models^{st} G'$.

The following proposition is a direct consequence of Proposition 7 and the above result, and shows that stable model entailment extends RDFS entailment from RDF graphs to ERDF ontologies.

Proposition 16. Let G, G' be RDF graphs such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$, $V_{G'} \cap \mathcal{V}_{ERDF} = \emptyset$, and $V_{G'} \cap sk_G(Var(G)) = \emptyset$. It holds: $G \models^{RDFS} G'$ iff $< G, \emptyset > \models^{st} G'$.

Recall that the Skolem vocabulary of G (that is, $sk_G(Var(G))$) contains artificial URIs giving "arbitrary" names to the anonymous entities whose existence was asserted by the use of blank nodes in G. Thus, the condition $V_{G'} \cap sk_G(Var(G)) = \emptyset$ in Proposition 16 is actually trivial.

Definition 23 (Query, Stable answers). Let $O = \langle G, P \rangle$ be an ERDF ontology. A *query* F is an ERDF formula. The *(ERDF) stable answers* of F w.r.t. O are defined as follows:

$$Ans^{st}_O(F) = \begin{cases} \text{“yes”} \ \text{ if } FVar(F) = \emptyset \text{ and } \forall M \in \mathcal{M}^{st}(M) : M \models F \\ \text{“no”} \ \text{ if } FVar(F) = \emptyset \text{ and } \exists M \in \mathcal{M}^{st}(M) : M \not\models F \\ \{v : FVar(F) \to V_O \mid \forall M \in \mathcal{M}^{st}(O) : M \models v(F)\} \quad \text{ if } FVar(F) \neq \emptyset, \end{cases}$$

where $v(F)$ is the formula F after replacing all the free variables x in F by $v(x)$. □

For example, let $p, q, c, s, o \in URI$, let $G = \{p(s, o), rdf{:}type(s, c), rdf{:}type(o, c)\}$, and let $P = \{q(?x, ?y) \leftarrow rdf{:}type(?x, c) \wedge rdf{:}type(?y, c) \wedge {\sim}p(?x, ?y)\}$. Then, the stable answers of $F = q(?x, ?y)$ w.r.t. $O = \langle G, P \rangle$ are $Ans^{st}_O(F) = \{(?x = o, ?y = o), (?x = s, ?y = s), (?x = o, ?y = s)\}$.

Let $O = \langle G, P \rangle$, where $G = \{rdf{:}type(p, erdf{:}TotalProperty), q(s, o)\}$ and $P = \{\neg p(?x, ?y) \leftarrow {\sim}p(?x, ?y)\}$. Then, $Ans^{st}_O(p(?x, ?y))= Ans^{st}_O({\sim}p(?x, ?y))= Ans^{st}_O(\neg p(?x, ?y)) = \emptyset$. This is because, in contrast to the above example, p is a total property. Thus, there is a stable model M of O such that $M \models v(p(?x, ?y) \wedge{\sim} \neg p(?x, ?y))$, and another stable model M' of O such that $M' \models v({\sim}p(?x, ?y) \wedge \neg p(?x, ?y))$, for all mappings $v : \{?x, ?y\} \to V_O$.

Consider the ERDF ontology O of the example (paper assignment) below Definition 21. Then[15], $Ans^{st}_O(assign(P1, R2)) =$"yes" and $Ans^{st}_O(assign(P2, R1)) =$"no". Though $Ans^{st}_O(assign(P2, R1)) =$ "*no*", that is $assign(P2, R1)$ is not satisfied by all stable models of O, there is a stable model (M_3) that satisfies $assign(P2, R1)$. Indeed the answers of the query $assign(?x, ?y)$ w.r.t. the stable models M_3 and M_4 are of particular interest since both M_3 and M_4 satisfy $allAssigned(Paper, Reviewer)$.

The following definition defines the credulous stable answers of a query w.r.t. an ERDF ontology, that is the answers of the query w.r.t the particular stable models of O.

Definition 24 (Credulous stable answers). Let $O = \langle G, P \rangle$ be an ERDF ontology. The *credulous (ERDF) stable answers* of a query F w.r.t. O are defined as follows:

[15] For brevity, the namespace *ex*: is ignored.

$$c\text{-}Ans_O^{st}(F) = \begin{cases} \text{“yes”} & \text{if } FVar(F) = \emptyset \text{ and } \exists M \in \mathcal{M}^{st}(M) : M \models F \\ \text{“no”} & \text{if } FVar(F) = \emptyset \text{ and } \forall M \in \mathcal{M}^{st}(M) : M \not\models F \\ \{Ans_M^{st}(F) \mid M \in \mathcal{M}^{st}(O) \text{ and } Ans_M^{st}(F) \neq \emptyset\} & \text{if } FVar(F) \neq \emptyset, \end{cases}$$

where $Ans_M^{st}(F) = \{v : FVar(F) \to V_O \mid M \models v(F)\}$. □

Continuing with the paper assignment example, consider the query $F = allAssigned(Paper, Reviewer)$. Then, although $Ans_O^{st}(F) =$“no”, it holds $c\text{-}Ans_O^{st}(F) =$“yes”, indicating that there is at least one desirable assignment of the papers $P1, P2, P3$ to reviewers $R1, R2, R3$.

Consider now the query $F = allAssigned(Paper, Reviewer) \wedge assign(?x, ?y)$. Then,

$$\begin{aligned} c\text{-}Ans_O^{st}(F) = \{ & \{(?x = P1, ?y = R2), (?x = P2, ?y = R1), (?x = P3, ?y = R3)\}, \\ & \{(?x = P1, ?y = R2), (?x = P2, ?y = R3), (?x = P3, ?y = R1)\}\}, \end{aligned}$$

indicating all possible desirable assignments of papers. Obviously, the credulous stable answers of a query F can provide alternative solutions, which can be useful in a range of applications, where alternative scenarios naturally appear.

Closing this section, we would like to indicate several differences of the ERDF stable model semantics w.r.t. first-order logic (FOL). First, in our semantics a *domain closure assumption* is made. This is due to the fact that the domain of every Herbrand interpretation of an ERDF ontology O is Res_O^H, that is the union of the vocabulary of O (V_O) and the set of XML values of the well-typed XML literals in V_O minus the well-typed XML literals. This implies that quantified variables always range in a closed domain. To understand the implications of this assumption, consider the ERDF graph G, where ($V' = \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$)

$$G = \{rdf{:}type(x, ex{:}c1) \mid x \in \{ex{:}c1, ex{:}c2\} \cup V' - \{rdf{:}\_i \mid i \in 1, 2, ...\}\}$$

Additionally, consider the ERDF program P, where

$$\begin{aligned} P = \{ & rdf{:}type(?x, ex{:}c1) \leftarrow rdf{:}type(?x, rdfs{:}ContainerMembershipProperty). \\ & rdf{:}type(?x, ex{:}c2) \leftarrow true.\}. \end{aligned}$$

Let $F = \forall ?x\ rdf{:}type(?x, ex{:}c2) \supset rdf{:}type(?x, ex{:}c1)$. It holds that $\langle G, P\rangle \models^{st} F$. However, $G \cup P \not\models^{FOL} F$. This is because, there is a FOL model M of $G \cup P$ with a domain D and a variable assignment $v{:}\{?x\} \to D$ such that $M, v \models rdf{:}type(?x, ex{:}c2)$ and $M, v \not\models rdf{:}type(?x, ex{:}c1)$.

Another difference is due to the fact that in the definition of the ERDF stable model semantics, only minimal Herbrand interpretations are considered. Let

$$\begin{aligned} G = \{ & ex{:}teaches(ex{:}Ann, ex{:}CS301), \quad ex{:}teaches(ex{:}Peter, ex{:}CS505), \\ & rdf{:}type(ex{:}CS505, ex{:}GradCourse)\}. \end{aligned}$$

Let $F = \forall ?x\ ex{:}teaches(ex{:}Peter, ?x) \supset rdf{:}type(?x, ex{:}GradCourse)$. Then, $\langle G, \emptyset\rangle \models^{st} F$. However, $G \not\models^{FOL} F$. This is because, there is a FOL model M of G with a domain D and a variable assignment $v{:}\{?x\} \to D$ such that $M, v \models ex{:}teaches(ex{:}Peter, ?x)$ and $M, v \not\models rdf{:}type(?x, ex{:}GradCourse)$. In other words, FOL makes an open-world assumption for *ex:teaches*. Note that the stable model conclusion F is non-monotonic, meaning that extending G to

G', F may no longer be satisfied by the ERDF ontology $\langle G', \emptyset \rangle$ (i.e. it is possible that $\langle G', \emptyset \rangle \not\models^{st} F$).

Consider now $G' = G \cup \{rdf{:}type(ex{:}teaches, erdf{:}TotalProperty)\}$. Then, similarly to FOL, it holds $O = \langle G', \emptyset \rangle \not\models^{st} F$. This is because now *ex:teaches* is a total property. Thus, there is a stable model M of O and a variable assignment v: $\{?x\} \rightarrow Res^H_O$ such that $M, v \models ex{:}teaches(ex{:}Peter, ?x)$ and $M, v \not\models rdf{:}type(?x, ex{:}GradCourse)$. In other worlds, now an open-world assumption is made for *ex:teaches*, as in FOL. Thus, there might exist a course taught by *ex:Peter*, even if it is not explicitly indicated so in G'.

Note that the previous ERDF graph G can also be seen as a Description Logic [12] A-Box A, where

$$A = \{teaches(Ann, CS301), teaches(Peter, CS505), GradCourse(CS505)\}$$

Consider a T-Box $T = \emptyset$. Since Description Logics (DLs) are a fragments of first-order logic, it holds that $L = \langle A, T \rangle \not\models^{DL} \forall teaches.GradCourse(Peter)$, meaning that L does not satisfy that all courses taught by Peter are graduate courses. An interesting approach for supporting non-monotonic conclusions in DLs is taken in [11], where *DLs of minimal knowledge and negation as failure* (MKNF-DLs) are defined, by extending DLs with two modal operators **K, A**. Intuitively, **K** expresses minimal knowledge and $\neg\mathbf{A}$ expresses weak negation. It holds that $L \models^{\text{MKNF-DL}} \forall \mathbf{K} teaches.\mathbf{K} GradCourse(Peter)$, expressing that all courses known to be taught by Peter are known to be undergraduate courses. Note that this conclusion is non-monotonic, thus it cannot be derived by "classical" DLs. However, compared to our theory, MKNF-DLs do not support rules and closed-world assumptions on properties (i.e., $\neg p(?x, ?y) \leftarrow {\sim}p(?x, ?y)$).

7 ERDF Model Theory as Tarski-style Model Theory

Tarski-style model theory is not limited to classical first-order models, as employed in the semantics of OWL. It allows various extensions, such as relaxing the bivalence assumption (e.g., allowing for partial models) or allowing higher-order models. It is also compatible with the idea of nonmonotonic inference, simply by not considering all models of a rule as being intended, but only those models that satisfy a certain criterion. Thus, the stable model semantics for normal and (generalized) extended logic programs, as defined in [16, 17, 20, 19], can be viewed as a Tarski-style model-theoretic semantics for nonmonotonic derivation rules.

A Tarski-style model theory is a triple $\langle L, \mathcal{I}, \models \rangle$, such that

1. L is a set of formulas, called *language*,
2. $\mathcal{I}$ is a set of interpretations, and
3. $\models$ is a relation between interpretations and formulas, called *model relation*.

For each Tarski-style model theory $\langle L, \mathcal{I}, \models \rangle$, we can define

- a notion of a derivation rule $G \leftarrow F$ where $F \in L$ is called "condition" and $G \in L$ is called "conclusion";
- $\text{DR}_L = \{G \leftarrow F : F, G \in L\}$, the set of derivation rules of L;

– a standard model operator

$$\mathcal{M}(KB) = \{I \in \mathcal{I} |\ I \models X,\ \forall X \in KB\}$$

where $KB \subseteq L \cup \mathrm{DR}_L$, is a set of formulas and/or derivation rules, called a *knowledge base.*

Notice that in this way we can define rules also for logics which do not contain an implication connective. This shows that the concept of a rule is more fundamental than, and independent of, the concept of implication.

Typically, in knowledge representation theories not all models of a knowledge base are *intended* models. Except from the standard model operator $\mathcal{M}$, there are also non-standard model operators, which do not provide all models of a knowledge base, but only a special subset that is supposed to capture its intended models according to some semantics.

A particularly important type of such an "intended model semantics" is obtained on the basis of some *information ordering* $\leq$, which allows to compare the information content of two interpretations $I_1, I_2 \in \mathcal{I}$: whenever $I_1 \leq I_2$, we say that I_2 *is more informative* than I_1. We define a Tarski-style model theory extended by an information ordering as a quadruple $\langle L,\ \mathcal{I},\ \models,\ \leq\rangle$, and call it an *information model theory.*

For any information model theory, we can define a number of natural non-standard model operators, such as the *minimal* model operator

$$\mathcal{M}^{min}(KB) = minimal_{\leq}(\mathcal{M}(KB))$$

and various refinements of it, like the *stable generated* models [16, 17, 20, 19].

For any given model operator $\mathcal{M}^x : \mathcal{P}(L \cup \mathrm{DR}_L) \to \mathcal{P}(\mathcal{I})$, knowledge base $KB \subseteq L \cup \mathrm{DR}_L$, and $F \in L$ we can define an entailment relation

$$KB \models^x F \quad \text{iff} \quad \forall I \in \mathcal{M}^x(KB),\ I \models F$$

For non-standard model operators, like minimal and stable models, this entailment relation is typically *nonmonotonic*, in the sense that for an extension $KB' \supseteq KB$ it may be the case that KB entails F, but KB' does not entail F.

Our (ERDF) stable model theory can be seen as a Tarski-style model theory, where $L = L(URI \cup \mathcal{LIT})$, $\mathcal{I}$ is the set of ERDF interpretations over any vocabulary $V \subseteq URI \cup \mathcal{LIT}$, and the model relation $\models$ is as defined in Definition 6. In our theory, the intended model operator ($\mathcal{M}^{st}$) assigns to each ERDF ontology a (possible empty) set of stable models (Definition 21).

8 Related Work

In this section, we briefly review extensions of web ontology languages with rules.

TRIPLE [36] is a rule language for the Semantic Web that is especially designed for querying and transforming RDF models (or contexts), supporting RDF and a subset of OWL Lite. Its syntax is based on F-Logic [23] and supports an important fragment of first-order logic. A triple is represented by a statement of the form $s[p \to o]$ and sets of statements sharing the same subject can be aggregated using molecules of the form $s[p_1 \to o_1; p_2 \to o_2; ....]$. All variables must be explicitly quantified, either existentially or universally. Arbitrary formulas can be used in the body, while the head of rules (consequent) are restricted to be atoms or conjunctions of molecules. An interesting and relevant feature of TRIPLE is the use of models to collect sets of related sentences. In particular, part of the semantics of the RDF(S) vocabulary is represented as pre-defined rules (and not as semantic conditions on interpretations), which are grouped together in a module. TRIPLE provides other features like path expressions, skolem model terms, as well as model intersection and difference. Finally, it should be mentioned that the queries and models are compiled into XSB prolog, which guarantees termination of inference. TRIPLE uses the Lloyd-Topor transformations [27] to take care of the first-order connectives in the sentences and supports negation-as-failure under the well-founded semantics [15]. Strong negation is not used.

Flora-2 [41] is a rule-based object-oriented knowledge base system for reasoning with semantic information on the Web. It is based on F-logic [23] and supports metaprogramming, nonmonotonic multiple inheritance, logical database updates, encapsulation, dynamic modules, and two kinds of weak negation (specifically, Prolog negation and well-founded negation [15]) through invocation of the corresponding operators \+ and *tnot* of the XSB system [32]). The formal semantics for nonmonotonic multiple inheritance is defined in [42]. In addition, Flora-2 supports reification and anonymous resources [43]. In particular, in Flora-2, reified statements $\$\{s(p \rightarrow o)\}\$$ are themselves objects. In contrast, in RDF(S), they are referred to by a URI or a blank node x, and are associated with the following RDF triples: $rdf{:}type(x, rdf{:}Statement)$, $rdf{:}subject(x, s)$, $rdf{:}predicate(x, p)$, and $rdf{:}object(x, o)$. In RDF(S) model theory (and thus, in our theory), no special semantics are given to reified statements. In Flora-2, anonymous resources are handled through skolemization (similarly to our theory).

Notation 3 (N3) provides a more human readable syntax for RDF and also extends RDF by adding numerous pre-defined constructs ("built-ins") for being able to express rules conveniently (see [38]). Remarkably, N3 contains a built-in (log:definitiveDocument) for making restricted completeness asumptions and another built-in (log:notIncludes) for expressing simple negations-as-failure tests. The addition of these constructs was motivated by use cases. However, N3 does not have any direct formal semantics for these constructs, and does not provide strong negation. Notation 3 is supported by the CWM system[16], a forward engine especially designed for the Semantic Web, and the Euler system[17], a backward engine relying on loop checking techniques to guarantee termination.

In [2], the authors propose the Paraconsistent Well-founded Semantics with explicit negation ($WFSX_P$)[18] [1], as the appropriate semantics for reasoning with (possibly, contradictory) information in the Semantic Web. Supporting arguments include (i) possible reasoning even in the presence of contradiction, (ii) program transformation into WFS, and (iii) polynomial time inference procedures. A particular implementation is the SEW system [8], which is able to reason with RDFS ontologies and rules (possibly with weak and strong negation),

[16] http://www.w3.org/2000/10/swap/doc/cwm.html

[17] http://www.agfa.com/w3c/euler/

[18] $WFSX_P$ is an extension of the well-founded semantics with explicit negation ($WFSX$) [31] on extended logic programs and, thus, also of the well-founded semantics (WFS) [15] on normal logic programs.

based on the $WFSX_P$ semantics. No formal model theory have been explicitly provided for the integrated logic.

DR-Prolog [4] and DR-DEVICE [6] are two systems that integrate RDFS ontologies with rules (strict or defeasible), that are partially ordered through a superiority relation, based on the semantics of defeasible logic [5, 28]. Defeasible logic supports only one kind of negation (strong negation) and allows to reason in the presence of contradiction and incomplete information. It supports monotonic and nonmonotonic rules, exceptions, default inheritance, and preferences. No formal model theory have been explicitly provided for the integrated logic.

OWL-DL [29] is an ontology representation language for the Semantic Web, that is a syntactic variant of the $\mathcal{SHOIN}(\mathbf{D})$ description logic and a decidable fragment of first-order logic. However, the need for extending the expressive power of OWL-DL with rules has initiated several studies, including the SWRL (Semantic Web Rule Language) proposal [22]. In [21], it is shown that this extension is in general undecidable. $\mathcal{AL}$-log [10] was one of the first efforts to integrate Description Logics with (safe) datalog rules, while achieving decidability. It considers the basic description logic $\mathcal{ALC}$ and imposes the constraint that only concept DL-atoms are allowed to appear in the body of the rules, whereas the heads of the rules are always non DL-atoms. Additionally, each variable appearing in a concept DL atom in the body of a rule has also to appear in a non DL-atom in the body or head of the rule. CARIN [26] provides a framework for studying the effects of combining the description logic $\mathcal{ALCNR}$ with (safe) datalog rules. In CARIN, both concept and role DL-atoms are allowed in the body of the rules. It is shown that the integration is decidable if rules are non-recursive, or certain combinations of constructors are not allowed in the DL component, or rules are *role-safe* (imposing a constraint on the variables of role DL atoms in the body of the rules)[19]. In [30], it was shown that the integration of a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base L with a disjunctive datalog program P is decidable, if P is DL-safe, that is, all variables in a rule occur in at least one non DL-atom in the body of the rule. In this work, in contrast to $\mathcal{AL}$-log and CARIN, no tableaux algorithms are employed for query answering but L is translated to an disjunctive logic program $DD(L)$ which is combined with P for answering ground queries.

In this category of works, entailment on the extended with rules DL is based on first-order logic, that is both the DL component and the logic program are viewed as a set of first-order logic statements. Thus, negation-as-failure, closed-world-assumptions, and non-monotonic reasoning cannot be supported. In contrast in our work, we support both weak and strong negation, and allow closed-world and open-world reasoning on a selective basis.

A different kind of integration is achieved in [13], where a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base L communicates with an extended logic program P (possibly with weak and strong negation), only through DL-query atoms in the body of the rules. In particular, the description logic component L is used for answering the augmented, with input from the logic program, queries appearing in the (possibly weakly negated) DL-query atoms, thus allowing flow of knowledge from P

[19] A rule is *role-safe* if at least one of the variables x, y of each role DL atom $R(x, y)$ in the body of the rule, appears in some body atom of a *base* predicate, where a *base predicate* is an ordinary predicate that appears only in facts or in rule bodies.

to L and vice-versa. In [13], the answer set semantics of $\langle L, P\rangle$ are defined which generalize the answer set semantics [17] of ordinary extended logic programs. Similarly, in [14], a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base L communicates with a normal logic program P (possibly with weak negation), through DL-query atoms in the body of the rules. The well-founded semantics of $\langle L, P\rangle$ are defined which generalize the well-founded semantics [15] of ordinary normal logic programs. Obviously, in [13, 14], derived information concerns only non DL-atoms (that can be possibly used as input to DL-query atoms). Thus, rule-based reasoning is supported only for non DL-atoms. In contrast, in our work, properties and classes appearing in the ERDF graphs can freely appear in the heads and bodies of the rules, allowing even the derivation of metalevel statements such as subclass and subproperty relationships, property transitivity, property and class totalness.

In [33], the semantics of a *disjunctive* $\mathcal{AL}$-log knowledge base is defined based on the answer set semantics [17], extending $\mathcal{AL}$-log [10]. A disjunctive $\mathcal{AL}$-log knowledge base is the integration of an $\mathcal{ALC}$ knowledge base with a (safe) extended disjunctive logic program that allows concept and role DL-atoms in the body of the rules (along with weak negation on non DL-atoms). Similarly to our case, in defining the disjunctive $\mathcal{AL}$-log semantics, only the grounded versions of the rules are considered (by instantiating variables with DL individuals). However in [33], rule-based reasoning is supported only for non DL-atoms and DL-atoms in the body of the rules express constraints (thus their weak negation is not allowed).

9 Conclusions

In this paper, we extended RDF graphs to ERDF graphs by allowing negative triples, and then to ERDF ontologies with the inclusion of derivation rules, allowing freely appearance of (meta)properties and (meta)classes in the body and head of the rules, all logical factors $\sim$ (weak negation), $\neg$ (strong negation), $\supset$ (material implication), $\wedge$, $\vee$, $\forall$, $\exists$ in the body of the rules, and strong negation $\neg$ in the head of the rules. Moreover, the RDF(S) vocabulary was extended with the terms $erdf{:}TotalProperty$ and $erdf{:}TotalClass$, for representing the metaclasses of total properties and total classes, respectively. We have defined ERDF formulas, ERDF interpretations, and ERDF entailment on ERDF formulas, showing that it extends RDFS entailment on RDF graphs.

We have developed the model-theoretic semantics of ERDF ontologies, called *ERDF stable model semantics*, showing that stable model entailment extends ERDF entailment on ERDF graphs, and thus it also extends RDFS entailment on RDF graphs. The ERDF stable model semantics is based on partial logic [19], which extends the answer set semantics on extended logic programs. We have shown that classical (boolean) Herbrand model reasoning is a special case of our semantics, when all properties are total. In this case, similarly to classical logic, an open-world assumption is made for all properties and classes and the two negations (weak and strong negation collapse). Allowing totalness of properties and classes to be declared on a selective basis and the explicit representation of closed-world assumptions (as derivation rules) enables the combination of open-world and closed-world reasoning in the same framework.

Future work concerns the support of datatype maps, including XSD datatypes, and the extension of the ERDF vocabulary to other useful ontological categories, possibly in accordance with [37]. Moreover, future work concerns defining a syntax for ERDF ontologies and implementation issues of our semantics.

Appendix A: RDF(S) semantics

For self-containment, in this Appendix, we review the definitions of simple, RDF, and RDFS interpretations, as well as the definitions of satisfaction of an RDF graph and RDFS entailment. For details, see [24, 18].

Definition 25 (Simple interpretation). A *simple interpretation* I of a vocabulary V consists of:

- A non-empty set of resources Res_I, called the *domain* or *universe* of I.
- A set of properties $Prop_I$.
- A vocabulary interpretation mapping $I_V : V \cap URI \rightarrow Res_I \cup Prop_I$.
- An extension mapping $PT_I : Prop_I \rightarrow \mathcal{P}(Res_I \times Res_I)$.
- A mapping $IL_I : V \cap \mathcal{TL} \rightarrow Res_I$.
- A set of literal values $LV_I \subseteq Res_I$, which contains $V \cap \mathcal{PL}$.

We define the mapping: $I : V \rightarrow Res_I \cup Prop_I$ such that:

- $I(x) = I_V(x), \ \forall x \in V \cap URI.$
- $I(x) = x, \ \forall\, x \in V \cap \mathcal{PL}.$
- $I(x) = IL_I(x), \ \forall\, x \in V \cap \mathcal{TL}.$ □

Definition 26 (Satisfaction of an RDF graph w.r.t. a simple interpretation). Let G be an *RDF* graph and let I be a simple interpretation of a vocabulary V. Let v be a mapping $v : Var(G) \rightarrow Res_I$. If $x \in Var(G)$, we define $[I + v](x) = v(x)$. If $x \in V$, we define $[I + v](x) = I(x)$. Then,

- $I, v \models G$ iff $\forall p(s, o) \in G, \ \ p \in V, \ s, o \in V \cup Var, \ I(p) \in Prop_I$, and $\langle [I + v](s), [I + v](o) \rangle \in PT_I(I(p))$.
- I *satisfies* the ERDF graph G, denoted by $I \models G$, iff there exists a mapping $v : Var(G) \rightarrow Res_I$ such that $I, v \models G$. □

Definition 27 (RDF interpretation). An *RDF interpretation* I of a vocabulary V is a simple interpretation of $V \cup \mathcal{V}_{RDF}$, which satisfies the following semantic conditions:

1. $x \in Prop_I$ iff $\langle x, I(rdf{:}Property) \rangle \in PT_I(I(rdf{:}type))$.
2. If "s"^^$rdf{:}XMLLiteral \in V$ and s is a well-typed XML literal string, then $IL_I($"s"^^$rdf{:}XMLLiteral)$ is the XML value of s, $IL_I($"s"^^$rdf{:}XMLLiteral) \in LV_I$, and $IL_I($"s"^^$rdf{:}XMLLiteral) \in CT_I(I(rdf{:}XMLLiteral))$.
3. If "s"^^$rdf{:}XMLLiteral \in V$ and s is an ill-typed XML literal string then $IL_I($"s"^^$rdf{:}XMLLiteral) \in Res_I - LV_I$, and $\langle IL_I($"s"^^$rdf{:}XMLLiteral), I(rdf{:}XMLLiteral) \rangle \notin PT_I(I(rdf{:}type))$.
4. I satisfies the *RDF* axiomatic triples, shown in Table 2. □

Definition 28 (RDF Entailment). Let G, G' be RDF graphs. We say that G *RDF-entails* G' ($G \models^{RDF} G'$) iff for every RDF interpretation I, if $I \models G$ then $I \models G'$. □

Definition 29 (RDFS interpretation). An *RDFS interpretation* I of a vocabulary V is an RDF interpretation of $V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS}$, extended by the new ontological category $Cls_I \subseteq Res_I$ for classes, as well as the class extension mapping $CT_I : Cls_I \rightarrow \mathcal{P}(Res_I)$, such that:

1. $x \in CT_I(y)$ iff $\langle x, y \rangle \in PT_I(I(rdf{:}type))$.

2. The ontological categories are defined as follows:
   $Cls_I = CT_I(I(rdfs{:}Class))$,
   $Res_I = CT_I(I(rdfs{:}Resource))$, and
   $LV_I = CT_I(I(rdfs{:}Literal))$.
3. if $\langle x, y\rangle \in PT_I(I(rdfs{:}domain))$ and $\langle z, w\rangle \in PT_I(x)$ then $z \in CT_I(y)$.
4. If $\langle x, y\rangle \in PT_I(I(rdfs{:}range))$ and $\langle z, w\rangle \in PT_I(x)$ then $w \in CT_I(y)$.
5. If $x \in Cls_I$ then $\langle x, I(rdfs{:}Resource)\rangle \in PT_I(I(rdfs{:}subclassOf))$.
6. If $\langle x, y\rangle \in PT_I(I(rdfs{:}subClassOf))$ then $x, y \in Cls_I$, $CT_I(x) \subseteq CT_I(y)$.
7. $PT_I(I(rdfs{:}subClassOf))$ is a reflexive and transitive relation on Cls_I.
8. If $\langle x, y\rangle \in PT_I(I(rdfs{:}subPropertyOf))$ then $x, y \in Prop_I$, $PT_I(x) \subseteq PT_I(y)$.
9. $PT_I(I(rdfs{:}subPropertyOf))$ is a reflexive and transitive relation on $Prop_I$.
10. If $x \in CT_I(I(rdfs{:}Datatype))$ then $\langle x, I(rdfs{:}Literal)\rangle \in PT_I(I(rdfs{:}subClassOf))$.
11. If $x \in CT_I(I(rdfs{:}ContainerMembershipProperty))$ then $\langle x, I(rdfs{:}member)\rangle \in PT_I(I(rdfs{:}subPropertyOf))$.
12. I satisfies the $RDFS$ axiomatic triples, shown in Table 3. □

Definition 30 (RDFS Entailment). Let G, G' be RDF graphs. We say that G *RDFS-entails* G' ($G \models^{RDFS} G'$) iff for every RDFS interpretation I, if $I \models G$ then $I \models G'$. □

Appendix B: Proofs

In this Appendix, we prove the Propositions presented in the main paper. To reduce the size of the proofs, we have eliminated the namespace from the URIs in $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$.

Proposition 1. Let F be an ERDF formula and let I be a partial interpretation of a vocabulary V. Let u, u' be mappings $u, u' : Var(F) \rightarrow Res_I$ such that $u(x) = u'(x)$, $\forall x \in FVar(F)$. It holds: $I, u \models F$ iff $I, u' \models F$.

Proof: We prove the proposition by induction. Without loss of generality, we assume that $\neg$ appears only in front of positive ERDF triples. Otherwise we apply the transformation rules of Definition 5, to get an equivalent formula that satisfies the assumption.

Let $F = p(s, o)$. It holds: $I, u \models F$ iff $p \in V'$, $s, o \in V' \cup Var$, $I(p) \in Prop_I$, and $\langle [I + u](s), [I + u](o) \rangle \in PT_I(I(p))$ iff $p \in V'$, $s, o \in V' \cup Var$, $I(p) \in Prop_I$, and $\langle [I + u'](s), [I + u'](o) \rangle \in PT_I(I(p))$ iff $I, u' \models p(s, o)$.

Let $F = \neg p(s, o)$. It holds: $I, u \models F$ iff $p \in V'$, $s, o \in V' \cup Var$, $I(p) \in Prop_I$, and $\langle [I + u](s), [I + u](o) \rangle \in PF_I(I(p))$ iff $p \in V'$, $s, o \in V' \cup Var$, $I(p) \in Prop_I$, and $\langle [I + u'](s), [I + u'](o) \rangle \in PF_I(I(p))$ iff $I, u' \models \neg p(s, o)$.

Assumption: Assume that the lemma holds for the subformulas of F.

We will show that the lemma holds also for F.

Let $F = \sim G$. It holds: $I, u \models F$ iff $I, u \models \sim G$ iff $V_G \subseteq V$ and $I, u \not\models G$ iff $V_G \subseteq V$ and $I, u' \not\models G$ iff $I, u' \models \sim G$ iff $I, u' \models F$.

Let $F = F_1 \wedge F_2$. It holds: $I, u \models F$ iff $I, u \models F_1 \wedge F_2$ iff $I, u \models F_1$ and $I, u \models F_2$ iff $I, u' \models F_1$ and $I, u' \models F_2$ iff $I, u' \models F_1 \wedge F_2$ iff $I, u' \models F$.

Let $F = \exists x\, G$. We will show that (i) if $I, u \models F$ then $I, u' \models F$ and (ii) if $I, u' \models F$ then $I, u \models F$.

(i) Let $I, u \models F$. Then, $I, u \models \exists x G$. Thus, there exists $u_1 : Var(G) \rightarrow Res_I$ s.t. $u_1(y) = u(y)$, $\forall y \in Var(G) - \{x\}$, and $I, u_1 \models G$. Let u_2 be a mapping $u_2 : Var(G) \rightarrow Res_I$ s.t. $u_2(y) = u'(y)$ and $u_2(x) = u_1(x)$. Since $u(z) = u'(z)$, $\forall z \in FVar(F)$ and $x \in FVar(G)$, it follows that $u_1(z) = u_2(z)$, $\forall z \in FVar(G)$. Thus, $I, u_2 \models G$. Therefore, there exists a mapping $u_2 : Var(G) \rightarrow Res_I$ s.t. $u_2(y) = u'(y)$, $\forall y \in Var(G) - \{x\}$, and $I, u_2 \models G$. Thus, $I, u' \models \exists x\, G$, which implies that $I, u' \models F$.

(ii) We prove this statement similarly to (i) by exchanging u and u'.

Let $F = F_1 \vee F_2$ or $F = F_1 \supset F_2$ or $F = \forall x G$. We can prove, similarly to the above cases, that $I, u \models F$ iff $I, u' \models F$. $\square$

Proposition 2. Let $G = \{t_1, ..., t_n\}$ be an $ERDF$ graph and let $Var(G) = \{x_1, ..., x_k\}$. Let F be the ERDF formula $\exists x_1, ...x_k\ t_1 \wedge ... \wedge t_n$. It holds: $I \models G$ iff $I \models F$.

Proof:

$\Rightarrow$) Assume that $I \models G$, we will show that $I \models F$. Since $I \models G$, it follows that $\exists v : Var(G) \rightarrow Res_I$ such that $I, v \models t_i$, $\forall i = 1, ...n$. Thus, $\exists v : Var(G) \rightarrow Res_I$ such that $I, v \models t_1 \wedge ... \wedge t_n$. This implies that, $\exists u : Var(G) \rightarrow Res_I$ such that $I, u \models F$. Since $FVar(F) = \emptyset$, it follows from Proposition 1 that $\forall u' : Var(G) \rightarrow Res_I$, it holds that $I, u' \models F$. Thus, $I \models F$.

$\Leftarrow$) Assume that $I \models F$, we will show that $I \models G$. Since $I \models F$, it follows that $\forall v : Var(G) \rightarrow Res_I$ it holds that $I, v \models F$. Thus, $\exists v : Var(G) \rightarrow Res_I$ such that $I, v \models F$. This implies that $\exists u : Var(G) \rightarrow Res_I$ such that $I, u \models t_1 \wedge ... \wedge t_n$. Thus, $\exists u : Var(G) \rightarrow Res_I$ such that $I, u \models t_i$, $\forall i = 1, ...n$. Therefore, $I \models G$. $\square$

Proposition 3. Let I be a coherent ERDF interpretation of a vocabulary V. It holds: $\forall x \in Cls_I$, $\ CT_I(x) \cap CF_I(x) = \emptyset$.

Proof: Since $I(type) \in Prop_I$, it holds: $PT_I(I(type)) \cap PF_I(I(type)) = \emptyset$. Assume that there is $x \in Cls_I$ s.t. $CT_I(x) \cap CF_I(x) \neq \emptyset$. Let $z \in CT_I(x) \cap CF_I(x)$, for such a x. Then, it holds $\langle z, x\rangle \in PT_I(I(type)) \cap PF_I(I(type))$, which is impossible. Thus, $\forall x \in Cls_I, \quad CT_I(x) \cap CF_I(x) = \emptyset$. □

Proposition 4. Let I be an ERDF interpretation of a vocabulary V and let $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Then,

1. For all $p, s, o \in V'$, such that $I(p) \in TProp_I$, it holds:
   $I \models \sim p(s, o)$ iff $I \models \neg p(s, o)$ (equivalently, $I \models p(s, o) \vee \neg p(s, o)$).
2. For all $x, c \in V'$ such that $I(c) \in TCls_I$, it holds:
   $I \models \sim rdf{:}type(x, c)$ iff $I \models \neg rdf{:}type(x, c)$
   (equivalently, $I \models rdf{:}type(x, c) \vee \neg rdf{:}type(x, c)$).

Proof:
1) It holds: $I \models \sim p(s, o)$ iff $I \not\models p(s, o)$ iff $\langle I(s), I(o)\rangle \notin PT_I(p)$ iff (since $p \in TProp_I$) $\langle I(s), I(o)\rangle \in PF_I(p)$ iff $I \models \neg p(s, o)$. Therefore, $I \models \sim p(s, o)$ iff $I \models \neg p(s, o)$.

We will also show that $I \models p(s, o) \vee \neg p(s, o)$. It holds $I \models p(s, o)$ or $I \models \sim p(s, o)$. This implies that $I \models p(s, o)$ or $I \models \neg p(s, o)$, and thus, $I \models p(s, o) \vee \neg p(s, o)$.
2) The proof is similar to the proof of 1) after replacing $p(s, o)$ by $type(x, c)$ and $TProp_I$ by $TCls_I$. □

Proposition 5. Let I be an ERDF interpretation of a vocabulary V and let $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Then,

1. If $TProp_I = Prop_I$ then I is a classical ERDF interpretation.
2. If I is a classical ERDF interpretation and F is an ERDF formula over V' such that $I(p) \in Prop_I$, for every property p in F, then it holds:
   $I \models \sim F$ iff $I \models \neg F$ (equivalently, $I \models F \vee \neg F$).

Proof:
1. If $TProp_I = Prop_I$ then for all $p \in Prop_I$, it holds that $PT_I(p) \cup PF_I(p) = Res_I \times Res_I$. Thus, I is a classical ERDF interpretation.
2. We will prove that $I \models \sim F$ iff $I \models \neg F$, by induction. Without loss of generality, we assume that $\neg$ appears only in front of positive ERDF triples. Otherwise we apply the transformation rules of Definition 5, to get an equivalent formula that satisfies the assumption.

Let $F = p(s, o)$. It holds: $I \models \sim F$ iff $\forall v : Var(F) \to Res_I$, it holds $I, v \not\models p(s, o)$ iff $\forall v : Var(F) \to Res_I$, it holds $\langle [I + v](s), [I + v](o)\rangle \notin PT_I(I(p))$ iff $\forall v : Var(F) \to Res_I$, it holds $\langle [I + v](s), [I + v](o)\rangle \in PF_I(I(p))$ iff $\forall v : Var(F) \to Res_I$, it holds $I, v \models \neg p(s, o)$ iff $I \models \neg F$.

Let $F = \neg p(s, o)$. It holds: $I \models \sim F$ iff $\forall v : Var(F) \to Res_I$, it holds $I, v \not\models \neg p(s, o)$ iff $\forall v : Var(F) \to Res_I$, it holds $\langle [I + v](s), [I + v](o)\rangle \notin PF_I(I(p))$ iff $\forall v : Var(F) \to Res_I$, it holds $\langle [I + v](s), [I + v](o)\rangle \in PT_I(I(p))$ iff $\forall v : Var(F) \to Res_I$, it holds $I, v \models \neg(\neg p(s, o))$ iff $I \models \neg F$.
Assumption: Assume that the lemma holds for the subformulas of F.
We will show that the lemma holds also for F.

Let $F = \sim G$. It holds: $I \models \sim F$ iff $\forall v : Var(F) \to Res_I$, it holds $I, v \not\models \sim G$ iff $\forall v : Var(F) \to Res_I$, it holds $I, v \models G$ iff $\forall v : Var(F) \to Res_I$, it holds $I, v \models \neg(\sim G)$ iff $I \models \neg F$.

Let $F = F_1 \wedge F_2$. It holds: $I \models \sim F$ iff $\forall v : Var(F) \to Res_I$, it holds $I, v \not\models F_1 \wedge F_2$ iff iff $\forall v : Var(F) \to Res_I$, it holds $I, v \not\models F_1$ or $I \not\models F_2$ iff (i) $\forall v : Var(F) \to Res_I$, it holds $I, v \models \sim F_1$ or (ii) $\forall v : Var(F) \to Res_I$, it holds $I, v \models \sim F_2$ iff (i) $\forall v : Var(F) \to Res_I$, it holds $I, v \models \neg F_1$ or (ii) $\forall v : Var(F) \to Res_I$, it holds $I, v \models \neg F_2$ iff $\forall v : Var(F) \to Res_I$, it holds $I, v \models \neg(F_1 \wedge F_2)$ iff $I \models \neg F$.

Let $F = \exists x\ G$. It holds: $I \models \sim F$ iff $\forall v : Var(F) \to Res_I$, it holds $I, v \models \sim\exists x G$ iff $\forall v : Var(F) \to Res_I$, there is no $u : Var(G) \to Res_I$ s.t. $u(y) = v(y), \quad \forall y \in Var(G) - \{x\}$ and $I, u \models G$ iff $\forall v : Var(F) \to Res_I$ and $\forall u : Var(G) \to Res_I$ s.t. $u(y) = v(y), \quad \forall y \in Var(G) - \{x\}$, it holds $I, u \models \sim G$ iff $\forall v : Var(F) \to Res_I$ and

$\forall u : Var(G) \to Res_I$ s.t. $u(y) = v(y),\ \ \forall y \in Var(G) - \{x\}$, it holds $I, u \models \neg G$ iff $\forall v : Var(F) \to Res_I,\ I, v \models \forall x\, \neg G$ iff $\forall v : Var(F) \to Res_I,\ I, v \models \neg \exists x\, G$ iff $I \models \neg F$.

Let $F = F_1 \vee F_2$ or $F = F_1 \supset F_2$ or $F = \forall x G$. We can prove, similarly to the above cases, that $I \models \sim\! F$ iff $I \models \neg F$. $\Box$

Proposition 6 Let G be an RDF graph such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$. Then, there is an RDFS interpretation that satisfies G iff there is an ERDF interpretation that satisfies G.

Proof:

$\Rightarrow$) Let I be an RDFS interpretation of a vocabulary V s.t. $I \models G$. In the proof of Proposition 7 ($\Leftarrow$), we show that we can construct an ERDF interpretation J of V such that $J \models G$.

$\Leftarrow$) Let I be an ERDF interpretation of a vocabulary V s.t. $I \models G$. In the proof of Proposition 7 ($\Rightarrow$), we show that we can construct an RDFS interpretation J of V such that $J \models G$. $\Box$

Proposition 7. Let G, G' be RDF graphs such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$ and $V_{G'} \cap \mathcal{V}_{ERDF} = \emptyset$. Then, $G \models^{RDFS} G'$ iff $G \models^{ERDF} G'$.

Proof: First, we define the set of *ERDF property classes*, $\mathcal{PC}_{ERDF} = \{TotalProperty, SymmetricProperty, TransitiveProperty\}$.

$\Leftarrow$) Let $G \models^{ERDF} G'$. We will show that $G \models^{RDFS} G'$. In particular, let I be an RDFS interpretation of a vocabulary V s.t. $I \models G$, we will show that $I \models G'$.

Since $I \models G$, it holds that $\exists v : Var(G) \to Res_I$ s.t. $I, v \models G$. Our goal is to construct an ERDF interpretation J of V s.t. $J \models G$. We consider an 1-1 mapping $res : \mathcal{V}_{ERDF} \to R$, where R is a set disjoint from Res_I. Additionaly, let $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDFS}$. Based on I and the mapping res, we construct a partial interpretation J of V as follows:

- $Res_J = Res_I \cup res(\mathcal{V}_{ERDF})$.
- $J_V(x) = I_V(x),\ \forall x \in (V' - \mathcal{V}_{ERDF}) \cap URI$ and $J_V(x) = res(x),\ \forall x \in \mathcal{V}_{ERDF}$.
- We define the mapping: $IL_J : V' \cap \mathcal{TL} \to Res_J$ such that: $IL_J(x) = IL_I(x)$.
- We define the mapping: $J : V' \to Res_J$ such that:
  - $J(x) = J_V(x),\ \forall x \in V' \cap URI$.
  - $J(x) = x,\ \forall\, x \in V' \cap \mathcal{PL}$.
  - $J(x) = IL_J(x),\ \forall\, x \in V' \cap \mathcal{TL}$.
- We define the mapping $PT'_J : Res_J \to \mathcal{P}(Res_J \times Res_J)$ as follows:

  (PT1) if $x, y, z \in Res_I$ and $\langle x, y\rangle \in PT_I(z)$ then $\langle x, y\rangle \in PT'_J(z)$.
  (PT2) $\langle res(TotalClass), J(Class)\rangle \in PT'_J(J(subClassOf))$.
  (PT3) if $x \in \mathcal{PC}_{ERDF}$ then $\langle res(x), J(Property)\rangle \in PT'_J(J(subClassOf))$.

  Starting from the derivations of (PT1), (PT2), and (PT3), the following rules are applied recursively, until a fixpoint is reached:

  (PT4) if $\langle x, y\rangle \in PT'_J(J(domain))$ and $\langle z, w\rangle \in PT'_J(x)$ then $\langle z, y\rangle \in PT'_J(J(type))$.
  (PT5) if $\langle x, y\rangle \in PT'_J(J(range))$ and $\langle z, w\rangle \in PT'_J(x)$ then $\langle w, y\rangle \in PT'_J(J(type))$.
  (PT6) if $\langle x, J(Class)\rangle \in PT'_J(J(type))$ then $\langle x, J(Resource)\rangle \in PT'_J(J(subClassOf))$.
  (PT7) if $\langle x, y\rangle \in PT'_J(J(subClassOf))$ then $\langle x, J(Class)\rangle \in PT'_J(J(type))$.
  (PT8) if $\langle x, y\rangle \in PT'_J(J(subClassOf))$ then $\langle y, J(Class)\rangle \in PT'_J(J(type))$.
  (PT9) if $\langle x, y\rangle \in PT'_J(J(subClassOf))$ and $\langle z, x\rangle \in PT'_J(J(type))$ then $\langle z, y\rangle \in PT'_J(J(type))$.
  (PT10) if $\langle x, J(Class)\rangle \in PT'_J(J(type))$ then $\langle x, x\rangle \in PT'_J(J(subClassOf))$.
  (PT11) if $\langle x, y\rangle \in PT'_J(J(subClassOf))$ and $\langle y, z\rangle \in PT'_J(J(subClassOf))$ then $\langle x, z\rangle \in PT'_J(J(subClassOf))$.
  (PT12) if $\langle x, y\rangle \in PT'_J(J(subPropertyOf))$ then $\langle x, J(Property)\rangle \in PT'_J(J(type))$.
  (PT13) if $\langle x, y\rangle \in PT'_J(J(subPropertyOf))$ then $\langle y, J(Property)\rangle \in PT'_J(J(type))$.

(PT14) if $\langle x, y\rangle \in PT'_J(J(subPropertyOf))$ and $\langle z, w\rangle \in PT'_J(x)$ then $\langle z, w\rangle \in PT'_J(y)$.
(PT15) if $\langle x, J(Property)\rangle \in PT'_J(J(type))$ then $\langle x, x\rangle \in PT'_J(J(subPropertyOf))$.
(PT16) if $\langle x, y\rangle \in PT'_J(J(subPropertyOf))$ and $\langle y, z\rangle \in PT'_J(J(subPropertyOf))$ then $\langle x, z\rangle \in PT'_J(J(subPropertyOf))$.
(PT17) if $\langle x, J(Datatype)\rangle \in PT'_J(J(type))$ then $\langle x, J(Literal)\rangle \in PT'_J(J(subClassOf))$.
(PT18) if $\langle x, J(ContainerMembershipProperty)\rangle \in PT'_J(J(type))$ then $\langle x, J(member)\rangle \in PT'_J(J(subPropertyOf))$.

After reaching fixpoint, nothing else is contained in $PT'_J(x)$, $\forall x \in Res_J$.

- $Prop_J = \{x \in Res_J \mid \langle x, J(Property)\} \in PT'_J(J(type))\}$.
- The mapping $PT_J : Prop_J \rightarrow \mathcal{P}(Res_J \times Res_J)$ is defined as follows: $PT_J(x) = PT'_J(x)$, $\forall x \in Prop_J$.
- $LV_J = \{x \in Res_J \mid \langle x, J(Literal)\rangle \in PT_J(J(type))\}$.
- The mapping $PF_J : Prop_J \rightarrow \mathcal{P}(Res_J \times Res_J)$ is defined as follows:

(PF1) if "s"^^$rdf{:}XMLLiteral \in V$ is not a well-typed XML-Literal then $\langle IL_J(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral), J(Literal)\rangle \in PF_J(J(type))$.
(PF2) if $\langle J(TotalClass), J(TotalClass)\rangle \in PT_J(J(type))$ then $\forall x \in Res_J - \{J(TotalClass)\}$, $\langle x, J(TotalClass)\rangle \in PF_J(J(type))$.
(PF3) if $\langle J(TotalProperty), J(TotalProperty)\rangle \in PT_J(J(type))$ then $\forall x, y \in Res_J$, $\langle x, y\rangle \in PF_J(J(TotalProperty))$.

Starting from the derivations of (PF1), (PF2), and (PF3), the following rules are applied recursively, until a fixpoint is reached:

(PF4) if $\langle x, y\rangle \in PT_J(J(subClassOf))$ and $\langle z, y\rangle \in PF_J(J(type))$ then $\langle z, x\rangle \in PF_J(type)$.
(PF5) if $\langle x, y\rangle \in PT_J(J(subPropertyOf))$ and $\langle z, w\rangle \in PF_J(y)$ then $\langle z, w\rangle \in PF_J(x)$.
(PF6) if $\langle J(SymmetricProperty), J(SymmetricProperty)\rangle \in PT_J(J(type))$ and $\langle x, y\rangle \in PF_J(J(SymmetricPropery))$ then $\langle y, x\rangle \in PF_J(J(SymmetricPropery))$

After reaching fixpoint, nothing else is contained in $PF_J(x)$, $\forall x \in Prop_J$.

Before we continue, we prove the following lemma:

Lemma: For all $x, y, x \in Res_J$, $\langle x, y\rangle \in PT'_J(z)$ iff $\langle x, y\rangle \in PT_J(z)$.
Proof:
$\Leftarrow$) if $\langle x, y\rangle \in PT_J(z)$, then from the definition of PT_J, it follows immediately that $\langle x, y\rangle \in PT'_J(z)$.
$\Rightarrow$) Let $\langle x, y\rangle \in PT'_J(z)$. Then, from the definition of PT'_J, it follows that it holds (i) $z \in Prop_I$ or (ii) $\exists w \in Res_J$, s.t. $\langle w, z\rangle \in PT'_J(J(subPropertyOf))$.

(i) Assume that $z \in Prop_I$. Then, $\langle z, I(Property)\rangle \in PT_I(I(type))$. This implies that $\langle z, J(Property)\rangle \in PT_I(J(type))$. ¿From (PT1), it now follows that $\langle z, J(Property)\rangle \in PT'_J(J(type))$. Therefore, $z \in Prop_J$. From the definition of PT_J, it now follows that $\langle x, y\rangle \in PT_J(z)$.

(ii) Assume that $\exists w \in Res_J$, s.t. $\langle w, z\rangle \in PT'_J(J(subPropertyOf))$. Then, from (PT13), it follows that $\langle z, J(Property)\rangle \in PT'_I(J(type))$. Therefore, $z \in Prop_J$. From the definition of PT_J, it now follows that $\langle x, y\rangle \in PT_J(z)$.
End of Lemma

Though not mentioned explicitly, the above Lemma is used throughout the rest of the proof.

To show that J is a partial interpretation of V', it is enough to show that $V' \cap \mathcal{PL} \subseteq LV_J$. Let $x \in V' \cap \mathcal{PL}$. Then, $x \in LV_I$. Thus, $\langle x, I(Literal)\rangle \in PT_I(I(type))$. Due to

(PT1), this implies that $\langle x, J(Literal)\rangle \in PT_J(J(type))$. Thus, $x \in LV_J$.

Now, we extend J with the ontological categories:
$Cls_J = \{x \in Res_J \mid \langle x, J(Class)\rangle \in PT_J(J(type))\}$,
$TCls_J = \{x \in Res_J \mid \langle x, J(TotalClass)\rangle \in PT_J(J(type))\}$, and
$TProp_J = \{x \in Res_J \mid \langle x, J(TotalProperty)\rangle \in PT_J(J(type))\}$.
We define $CT_J, CF_J : Cls_J \to \mathcal{P}(Res_J)$ as follows:
$x \in CT_J(y)$ iff $\langle x, y\rangle \in PT_J(J(type))$, and
$x \in CF_J(y)$ iff $\langle x, y\rangle \in PF_J(J(type))$.

We will now show that J is an ERDF interpretation of V. Specifically, we will show that J satisfies the semantic conditions of Definition 8 (ERDF Interpretation) and Definition 9 (Coherent ERDF interpretation).

First, we will show that J satisfies semantic condition 2 of Definition 8. We will start by proving that $Res_J = CT_J(J(Resource))$. Obviously, $CT_J(J(Resource)) \subseteq Res_J$. Thus, it is enough to prove that $Res_J \subseteq CT_J(J(Resource))$. Let $x \in Res_J$. Then, we distinguish the following cases:

Case 1) $x \in Res_I$. Since I is an RDFS interpretation, $\langle x, I(Resource)\rangle \in PT_I(I(type))$. Thus, $\langle x, J(Resource)\rangle \in PT_J(J(type))$, which implies that $x \in CT_J(J(Resource))$.

Case 2) $x \in res(\mathcal{V}_{ERDF})$. From the definition of PT'_J, it follows that $\langle x, J(Resource)\rangle \in PT'_J(J(type))$. Thus, $\langle x, J(Resource)\rangle \in PT_J(J(type))$, which implies that $x \in CT_J(J(Resource))$.

Thus, $Res_J = CT_J(J(Resource))$.

Additionally, it is easy to see that it holds $Prop_J = CT_J(J(Property))$, $Cls_J = CT_J(J(Class))$, $LV_J = CT_J(J(Literal))$, $TCls_J = CT_J(J(TotalClass))$, and $TProp_J = CT_J(J(TotalProperty))$.

We will now show that J satisfies semantic condition 3 of Definition 8. Let $\langle x, y\rangle \in PT_J(J(domain))$ and $\langle z, w\rangle \in PT_J(x)$. Then, from (PT4) and the definition of CT_J, it follows that $z \in CT_J(y)$.

We will now show that J satisfies semantic condition 4 of Definition 8. Let $\langle x, y\rangle \in PT_J(J(range))$ and $\langle z, w\rangle \in PT_J(x)$. Then, from (PT5) and the definition of CT_J, it follows that $w \in CT_J(y)$.

We will now show that J satisfies semantic condition 5 of Definition 8. Let $x \in Cls_J$. Thus, it holds: $\langle x, J(Class)\rangle \in PT_J(J(type))$. From (PT6), it now follows that $\langle x, J(Resource)\rangle \in PT_J(J(subClassOf))$.

We will now show that J satisfies semantic condition 6 of Definition 8. Let $\langle x, y\rangle \in PT_J(J(subClassOf))$. Then, from (PT7), (PT8), and the definition of CT_J, it follows that $x, y \in Cls_J$.

Let $\langle x, y\rangle \in PT_J(J(subClassOf))$. We will show that $CT_J(x) \subseteq CT_J(y)$. In particular, let $z \in CT_J(x)$. Then, from (PT9) and the definition of CT_J, it follows that $z \in CT_J(y)$.

Let $\langle x, y\rangle \in PT_J(J(subClassOf))$. We will show that $CF_J(y) \subseteq CF_J(x)$. In particular, let $z \in CF_J(y)$. Then, from (PF4) and the definition of CF_J, it follows that $z \in CF_J(x)$.

In a similar manner, we can prove that J also satisfies the semantic conditions 7, 8, 9, 10, and 11 of Definition 8.

To continue the rest of the proof, we need to make a few observations.

Consider the mapping $h : Res_J \rightarrow Res_I$, which is defined as follows:

$$h(x) = \begin{cases} x & \text{if } x \in Res_I \\ I(Class) & \text{if } x = res(TotalClass) \\ I(Property) & \text{if } x \in res(\mathcal{PC}_{ERDF}) \end{cases}$$

Observation 1: If $\langle x, y \rangle \in PT_J(z)$ and $y \in res(\mathcal{V}_{ERDF})$ then $x = y$.
Observation 2: If $x \in res(\mathcal{V}_{ERDF})$ and $x \in Prop_J$ then $PT_J(x) = \emptyset$.
Observation 3: If $\langle x, y \rangle \in PT_J(z)$ then $\langle h(x), h(y) \rangle \in PT_I(h(z))$.
Observation 4: If $x, y, z \in Res_I$ and $\langle x, y \rangle \in PT_J(z)$ then $\langle x, y \rangle \in PT_I(z)$[20].
The proof of these observations is made by induction. It is easy to see that all observations hold for the derivations of (PT1), (PT2), and (PT3). Assume now that the observations hold for the derivations obtained at a step k of the application of the fixpoint operator for PT_J. Then, the observations also hold for the derivations obtained at step $k + 1$.

We will now show that J satisfies semantic condition 12 of Definition 8. Let $x \in TCls_J$. Thus, $\langle x, J(TotalClass) \rangle \in PT_J(J(type))$. From *Observation 1*, it follows that $x = J(TotalClass)$. From (PF2), it now follows that $CT_J(J(TotalClass)) \cup CF_J(J(TotalClass)) = Res_J$. Thus, $CT_J(x) \cup CF_J(x) = Res_J$.

We will now show that J satisfies semantic condition 13 of Definition 8. Let $x \in TProp_J$. Thus, $\langle x, J(TotalProperty) \rangle \in PT_J(J(type))$. ¿From *Observation 1*, it follows that $x = J(TotalProperty)$. From (PF3), it now follows that $PT_J(J(TotalProperty)) \cup PF_J(J(TotalProperty)) = Res_J \times Res_J$. Thus, $PT_J(x) \cup PF_J(x) = Res_J \times Res_J$.

We will now show that J satisfies semantic condition 14 of Definition 8. Let $x \in CT_J(J(SymmetricProperty))$. Then, $\langle x, J(SymmetricProperty) \rangle \in PT_J(J(type))$. ¿From *Observation 1*, it follows that $x = J(SymmetricProperty)$. From *Observation 2*, it follows that $PT_J(x) = \emptyset$. Thus, $PT_J(x)$ is a symmetric relation. Additionally, from (PF6), it follows that $PF_J(x)$ is a symmetric relation.

We will now show that J satisfies semantic condition 15 of Definition 8. Let $x \in CT_J(J(TransitiveProperty))$. Then, $\langle x, J(TransitiveProperty) \rangle \in PT_J(J(type))$. ¿From *Observation 1*, it follows that $x = J(TransitiveProperty)$. From *Observation 2*, it follows that $PT_J(x) = \emptyset$. Thus, $PT_J(x)$ is a transitive relation.

We will now show that J satisfies semantic condition 16 of Definition 8. Let "s"^^$rdf{:}XMLLiteral$ be a well-typed XML-Literal in V then $IL_J($"s"^^$rdf{:}XMLLiteral) = IL_I($"$s$"^^$rdf{:}XMLLiteral)$ is the XML value of s. Additionally, since I is an RDFS interpretation of V, it holds: $\langle IL_I($"s"^^$rdf{:}XMLLiteral), I(XMLLiteral) \rangle \in PT_I(I(type))$. Therefore, from (PT1), it follows that
$\langle IL_J($"s"^^$rdf{:}XMLLiteral), J(XMLLiteral) \rangle \in PT_J(J(type))$.

We will now show that J satisfies semantic condition 17 of Definition 8. Let "s"^^$rdf{:}XMLLiteral \in V$ s.t s is not a well-typed XML literal string. Assume that $IL_J($"s"^^$rdf{:}XMLLiteral) \in LV_J$. Then, $\langle IL_J($"s"^^$rdf{:}XMLLiteral), J(Literal) \rangle \in PT_J(J(type))$. ¿From *Observation 4*, it follows that $\langle IL_J($"s"^^$rdf{:}XMLLiteral)$, $J(Literal) \rangle \in PT_I(J(type))$. Therefore, it follows that $\langle IL_I($"s"^^$rdf{:}XMLLiteral)$, $I(Literal) \rangle \in PT_I(I(type))$. Thus, $IL_I($"s"^^$rdf{:}XMLLiteral) \in LV_I$, which is impossible since I is an RDFS interpretation of V. Therefore, $IL_J($"s"^^$rdf{:}XMLLiteral) \in Res_J - LV_J$.

[20] Note that *Observation 3* implies *Observation 4*.

Additionally, from (PF1), it follows that $\langle IL_J(\text{“}s\text{”\^\^}rdf{:}XMLLiteral), J(Literal)\rangle \in PF_J(J(type))$.

J also satisfies semantic condition 18 of Definition 8, due to (PT1). Finally, J satisfies semantic condition 19, due to (PT2) and (PT3).

Thus, J is an ERDF interpretation of V.

Now, we will show that J is a coherent ERDF interpretation (Definition 9). Assume that this is not the case. Thus, there is $z \in Prop_J$ s.t. $PT_J(z) \cap PF_J(z) \neq \emptyset$. Assume that $\langle x, y\rangle \in PT_J(z) \cap PF_J(z)$, for such a z. We distinguish the following cases:

Case 1) $z \in res(\mathcal{V}_{ERDF})$. Then, from *Observation 2*, it follows that $PT_J(z) = \emptyset$, which is a contradiction.

Case 2) $y \in res(\mathcal{V}_{ERDF})$ and $z \in Res_I$. Then, it holds:
(i) $\langle z, res(TotalProperty)\rangle \in PT_J(J(subPropertyOf))$, or
(ii) $\langle z, J(type)\rangle \in PT_J(J(subPropertyOf))$ and $\langle x, y\rangle \in PF_J(J(type))$.
¿From *Observation 1* and since $z \in Res_I$, case (i) is impossible. Thus, $\langle z, J(type)\rangle \in PT_J(J(subPropertyOf))$ and $\langle x, y\rangle \in PF_J(J(type))$. This implies that
$y = res(TotalClass)$. From *Observation 1*, it follows that $x = res(TotalClass)$, which is impossible since, due to (PF2), $\langle res(TotalClass), res(TotalClass)\rangle \notin PF_J(J(type))$.

Case 3) $x \in res(\mathcal{V}_{ERDF})$ and $y, z \in Res_I$. Then, it holds:
(i) $\langle z, res(TotalProperty)\rangle \in PT_J(J(subPropertyOf))$, or
(ii) $\langle z, J(type)\rangle \in PT_J(J(subPropertyOf))$ and $\langle x, y\rangle \in PF_J(J(type))$.
¿From *Observation 1* and since $z \in Res_I$, case (i) is impossible. Thus, $\langle z, J(type)\rangle \in PT_J(J(subPropertyOf))$ and $\langle x, y\rangle \in PF_J(J(type))$. This implies that
$y = res(TotalClass)$, which is impossible, since $y \in Res_I$.

Case 4) $x, y, z \in Res_I$. Then, $x = IL_J(s)$, where s is an ill-typed XML-Literal in V, $\langle z, J(type)\rangle \in PT_J(J(subPropertyOf))$ and $\langle y, J(Literal)\rangle \in PT_J(J(subClassOf))$. Since $\langle x, y\rangle \in PT_J(z)$, it follows that $\langle x, y\rangle \in PT_J(J(type))$. Since $\langle y, J(Literal)\rangle \in PT_J(J(subClassOf))$, it follows that $\langle x, J(Literal)\rangle \in PT_J(J(type))$. ¿From *Observation 4*, it follows that $\langle IL_J(s), J(Literal)\rangle \in PT_I(J(type))$. Therefore,
$\langle IL_I(s), I(Literal)\rangle \in PT_I(I(type))$. But this implies that $IL_I(s) \in LV_I$, which is impossible since I is an RDFS interpretation of V.

Since all cases lead to contradiction, it follows that:
$\forall z \in Prop_J, \quad PT_J(z) \cap PF_J(z) = \emptyset$.

We will now show that $J, v \models G$. Let $p(s, o) \in G$. Since $I, v \models G$, it holds that $p \in V'$, $s, o \in V' \cup \mathit{Var}$. Note that, due to (PT1), it holds $Prop_I \subseteq Prop_J$. Since $p \notin \mathcal{V}_{ERDF}$, it holds $J(p) = I(p) \in Prop_I \subseteq Prop_J$. Since $s, o \notin \mathcal{V}_{ERDF}$, it holds that $[I+v](s) = [J+v](s)$ and $[I+v](o) = [J+v](o)$. Since $I, v \models G$, it holds $\langle [I+v](s), [I+v](o)\rangle \in PT_I(I(p))$. Thus, $\langle [J+v](s), [J+v](o)\rangle \in PT_I(J(p))$. From (PT1), it follows that $\langle [J+v](s), [J+v](o)\rangle \in PT_J(J(p))$. Thus, $J, v \models G$, which implies that $J \models G$. Since J is an ERDF interpretation and $G \models^{ERDF} G'$, it follows that $J \models G'$. Thus, there is $u : Var(G') \to Res_J = Res_I \cup res(\mathcal{V}_{ERDF})$ s.t. $J, u \models G'$. We define a mapping $u' : Var(G') \to Res_I$ as follows:

$$u'(x) = \begin{cases} u(x) & \text{if } u(x) \in Res_I \\ I(Class) & \text{if } u(x) = res(TotalClass) \\ I(Property) & \text{if } u(x) \in res(\mathcal{PC}_{ERDF}) \end{cases}$$

We will show that $I, u' \models G'$. Let $p(s, o) \in G'$. Since $J \models G'$ and $V_{G'} \cap \mathcal{V}_{ERDF} = \emptyset$, it follows that $p \in V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS}$, $s, o \in V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathit{Var}$, and $J(p) \in Prop_J$. Thus, $\langle J(p), J(type)\rangle \in PT_J(J(Property)$, which implies (since $p \notin \mathcal{V}_{ERDF}$) that $\langle I(p), I(type)\rangle \in PT_J(I(Property)$. Due to *Observation 4*, it follows that $\langle I(p), I(type)\rangle \in PT_I(I(Property)$. Thus, $I(p) \in Prop_I$. Additionally, it holds:

$\langle [J+u](s), [J+u](o)\rangle \in PT_J(J(p))$. We want to show that $\langle [I+u'](s), [I+u'](o)\rangle \in PT_I(I(p))$.

Case 1) It holds: (i) if $s \in Var(G')$ then $u(s) \notin res(\mathcal{V}_{ERDF})$ and (ii) if $o \in Var(G')$ then $u(o) \notin res(\mathcal{V}_{ERDF})$.
Then, $[J+u](s) = [J+u'](s) = [I+u'](s) \in Res_I$, $[J+u](o) = [J+u'](o) = [I+u'](o) \in Res_I$, and $J(p) = I(p) \in Res_I$. Thus, $\langle [J+u](s), [J+u](o)\rangle \in PT_J(J(p))$ implies that $\langle [I+u'](s), [I+u'](o)\rangle \in PT_J(I(p))$. From *Observation 4*, the latter implies that $\langle [I+u'](s), [I+u'](o)\rangle \in PT_I(I(p))$.

Case 2) It holds: (i) $s \in Var(G')$ and $u(s) \in res(\mathcal{V}_{ERDF})$ and (ii) if $o \in Var(G')$ then $u(o) \notin res(\mathcal{V}_{ERDF})$.
Assume that $u(s) = res(TotalClass)$, $[J+u](o) = y$, and $J(p) = z$. Then $y, z \in Res_I$. Additionally, $I(p) = J(p) = z$ and $[I+u'](o) = [J+u](o) = y$. Thus, $\langle [I+u'](s), [I+u'](o)\rangle = \langle I(Class), y\rangle$. It holds $\langle res(TotalClass), y\rangle \in PT_J(z)$. Due to *Observation 3*, it holds $\langle I(Class), y\rangle \in PT_I(z)$. Thus, $\langle [I+u'](s), [I+u'](o)\rangle = \langle I(Class), y\rangle \in PT_I(z) = PT_I(I(p))$.

Similarly, if $u(s) \in res(\mathcal{PC}_{ERDF})$, we prove that $\langle [I+u'](s), [I+u'](o)\rangle \in PT_I(I(p))$.

Case 3) It holds: $o \in Var(G')$ and $u(o) \in res(\mathcal{V}_{ERDF})$. Then, *Observation 1*, it follows that $s \in Var(G')$ and $u(s) = u(o)$. Assume that $u(o) = res(TotalClass)$, and $J(p) = z$. Then, $z \in Res_I$ and $I(p) = J(p) = z$. Additionally, $\langle [I+u'](s), [I+u'](o)\rangle = \langle I(Class), I(Class)\rangle$. It holds $\langle res(TotalClass), res(TotalClass)\rangle \in PT_J(z)$. Due to *Observation 3*, it follows that $\langle I(Class), I(Class)\rangle \in PT_I(z)$. Thus, $\langle [I+u'](s), [I+u'](o)\rangle = \langle I(Class), I(Class)\rangle \in PT_I(z) = PT_I(I(p))$.

Similarly, if $u(o) \in res(\mathcal{PC}_{ERDF})$, we prove that $\langle [I+u'](s), [I+u'](o)\rangle \in PT_I(I(p))$.

As in all cases, it holds $\langle [I+u'](s), [I+u'](o)\rangle = PT_I(I(p))$, it follows that $I, u' \models G'$, which implies that $I \models G'$.

$\Rightarrow$) Let $G \models^{RDFS} G'$. We will show that $G \models^{ERDF} G'$. Let I be an ERDF interpretation of a vocabulary V, such that $I \models G$. Thus, there is $u : Var(G) \rightarrow Res_I$ s.t. $I, u \models G$. We will show that $I \models G'$.

We define $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Based on I, we construct an RDFS interpretation J of V' such that: $Res_J = Res_I$, $Prop_J = Prop_I$, $LV_J = LV_I$, $Cls_J = Cls_I$, $J_V(x) = I_V(x), \forall x \in V' \cap URI$, $PT_J(x) = PT_I(x), \forall x \in Prop_J$, $IL_J(x) = IL_I(x), \forall x \in V' \cap \mathcal{TL}$, $CT_J(x) = CT_I(x), \forall x \in Cls_J$.

We will now show that J is indeed an RDFS interpretation of V'.

First, we will show that J satisfies semantic condition 1 of Definition 27 (Appendix, RDF interpretation). It holds: $x \in Prop_J$ iff $x \in Prop_I$ iff $x \in CT_I(I(Property))$ iff $\langle x, I(Property)\rangle \in PT_I(I(type))$ iff $\langle x, J(Property)\rangle \in PT_J(J(type))$.

We will show that J satisfies semantic condition 2 of Definition 27. Let "s"^^$rdf{:}XMLLiteral \in V$ such that s is a well-typed XML literal string. Then, it follows from the definition of J and the fact that I is an ERDF interpretation of V that $IL_J($"s"^^$rdf{:}XMLLiteral)$ is the XML value of s, and $IL_J($"s"^^$rdf{:}XMLLiteral) \in CT_J(J(XMLLiteral))$. We will show that $IL_J($"s"^^$rdf{:}XMLLiteral) \in LV_J$. Since I is an ERDF interpretation, $IL_I($"s"^^$rdf{:}XMLLiteral) \in CT_I(I(XMLLiteral))$. Additionally, $\langle I(XMLLiteral), I(Literal)\rangle \in PT_I(I(subClassOf))$. Therefore, $IL_I($"s"^^$rdf{:}XMLLiteral) \in CT_I(I(Literal))$, and thus, $IL_I($"s"^^$rdf{:}XMLLiteral) \in LV_I$. The last statement implies that $IL_J($"s"^^$rdf{:}XMLLiteral) \in LV_J$.

We will show that J satisfies semantic condition 3 of Definition 27. Let "s"^^$rdf{:}XMLLiteral \in V$ such that s is an ill-typed XML literal string. Then, it follows from the definition of J and the fact that I is an ERDF interpretation of V that $IL_J($"s"^^$rdf{:}XMLLiteral) \in Res_J - LV_J$. We will show that $\langle IL_J($"s"^^$rdf{:}XMLLiteral), J(XMLLiteral)\rangle \notin PT_J(J(type))$. Assume that $\langle IL_J($"s"^^$rdf{:}XMLLiteral), J(XMLLiteral)\rangle \in PT_J(J(type))$. Then,

$\langle IL_I(\text{“}s\text{”\^\^}rdf{:}XMLLiteral), I(XMLLiteral)\rangle \in PT_I(I(type))$. Thus, $IL_I(\text{“}s\text{”\^\^}rdf{:}XMLLiteral) \in CT_I(I(XMLLiteral))$. Since it holds $\langle I(XMLLiteral), I(Literal)\rangle \in PT_I(I(subClassOf))$, it follows that $IL_I(\text{“}s\text{”\^\^}rdf{:}XMLLiteral) \in CT_I(I(Literal))$. Thus, $IL_I(\text{“}s\text{”\^\^}rdf{:}XMLLiteral) \in LV_I$, which is impossible since I is an ERDF interpretation of V. Therefore, $\langle IL_J(\text{“}s\text{”\^\^}rdf{:}XMLLiteral), J(XMLLiteral)\rangle \notin PT_J(J(type))$.

It is easy to see that J satisfies semantic condition 4 of Definition 27 and all the semantic conditions of Definition 29 (Appendix A, RDFS Interpretation). Therefore, J is an RDFS interpretation of V'.

We will now show that $J, u \models G$. Let $p(s, o) \in G$. Since $I \models G$, it holds that $p \in V'$, $s, o \in V' \cup Var$, and $J(p) = I(p) \in Prop_I = Prop_J$. It holds: $\langle [J + u](s), [J + u](o)\rangle \in PT_J(J(p))$ iff $\langle [I + u](s)), [I + u](o)\rangle \in Prop_I(I(p))$, which is true, since $I, u \models G$. Thus, $J, u \models G$, which implies that $J \models G$. Since $G \models^{RDFS} G'$, it follows that $J \models G'$. Thus, there is $v : Var(G') \to Res_J$ s.t. $J, v \models G'$.

We will now show that $I \models G'$. Let $p(s, o) \in G'$. Since $J, v \models G'$, it holds that $p \in V'$, $s, o \in V' \cup Var$, and $I(p) = J(p) \in Prop_J = Prop_I$. It holds: $\langle [I + v](s), [I + v](o)\rangle \in PT_I(I(p))$ iff $\langle [J+v](s), [J+v](o)\rangle \in PT_J(J(p))$, which is true, since $J, v \models G'$. Thus, $I, v \models G'$, which implies that $I \models G'$. $\square$

Proposition 8. Let G be an ERDF graph. There is an ERDF interpretation that satisfies G iff there is an ERDF interpretation that satisfies $sk(G)$.

Proof:
$\Rightarrow$) Let I be an ERDF interpretation of a vocabulary V such that $I \models G$. We will show that there is an ERDF interpretation J s.t. $J \models sk(G)$. Since $I \models G$, there is a total function $u : Var(G) \to Res_I$ s.t. $I, u \models G$. We define $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{RDFS}$. We construct an ERDF interpretation J of $V \cup sk_G(Var(G))$ as follows: $Res_J = Res_I$, $Prop_J = Prop_I$, $LV_J = LV_I$, $Cls_J = Cls_I$. We define $J_V : (V' \cup sk_G(Var(G))) \cap URI \to Res_J$, as follows: $J_V(x) = I_V(x), \forall x \in V' \cap URI$ and $J_V(x) = u(sk_G^{-1}(x)), \forall x \in sk_G(Var(G))$. Moreover, $PT_J(x) = PT_I(x), \forall x \in Prop_J$, $IL_J(x) = IL_I(x), \forall x \in V' \cap \mathcal{TL}$, $CT_J(x) = CT_I(x), \forall x \in Cls_J$.

Since I is an ERDF interpretation of V, it is easy to see that J is indeed an ERDF interpretation of $V \cup sk_G(Var(G))$. We will show that $J \models sk(G)$. First, we define a total function $g : V' \cup sk_G(Var(G)) \to V' \cup Var(G)$ as follows: $g(x) = sk_G^{-1}(x)$, $\forall x \in sk_G(Var(G))$ and $g(x) = x$, otherwise. Let $p(s, o) \in sk(G)$. Since $I \models G$, it follows that $p \in V'$, $s, o \in V' \cup Var$, and $J(p) = I(p) \in Prop_I = Prop_J$. It holds $J(s) = [I+u](g(s))$, $J(s) = [I + u](g(s))$, and $J(p) = I(p)$. Therefore, it holds: $\langle J(s), J(o)\rangle \in PT_J(J(p))$ iff $\langle [I + u](g(s)), [I + u](g(o))\rangle \in PT_I(I(p))$, which holds since $p(g(s), g(o)) \in G$ and $I, u \models G$. Therefore, $J \models sk(G)$.

$\Leftarrow$) It follows directly from Proposition 9. $\square$

Proposition 9. Let G be an ERDF graph and let I be an ERDF interpretation. Then, $I \models sk(G)$ implies $I \models G$.

Proof: Let I be an ERDF interpretation of a vocabulary V, such that $I \models sk(G)$. We will show that I satisfies G. We define $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Additionally, we define a total function $u : Var(G) \to Res_I$ s.t. $u(x) = I_V(sk_G(x)), \forall x \in Var(G)$. Moreover, we define a total function $u' : V' \cup Var(G) \to V'$ s.t. $u'(x) = sk_G(x)$, if $x \in Var(G)$ and $u'(x) = x$, otherwise. It is enough to show that $I, u \models G$. Let $p(s, o) \in G$. Then, $p \in V'$, $s, o \in V' \cup Var$, and $I(p) \in Prop_I$. It holds: $\langle [I + u](s), [I + u](o)\rangle \in PT_I(I(p))$ iff $\langle I(u'(s)), I(u'(o))\rangle \in Prop_I(I(p))$, which is true, since $p(u'(s), u'(o)) \in sk(G)$ and $I \models sk(G)$. Thus, $I, u \models G$, which implies that $I \models G$. $\square$

Proposition 10. Let G be an ERDF graph and F be an ERDF formula such that $V_F \cap sk_G(Var(G)) = \emptyset$. It holds: $G \models^{ERDF} F$ iff $sk(G) \models^{ERDF} F$.
Proof:
$\Rightarrow$) Let $G \models^{ERDF} F$. We will show that $sk(G) \models^{ERDF} F$. Let I be an ERDF interpretation over a vocabulary V s.t. $I \models sk(G)$. Then by Proposition 9, it follows that $I \models G$. Since $G \models^{ERDF} F$, it follows that $I \models F$.

$\Leftarrow$) Let $sk(G) \models^{ERDF} F$. We will show that $G \models^{ERDF} F$. Let I be an ERDF interpretation of a vocabulary V such that $I \models G$. We will show that $I \models F$. In the proof of Proposition 8, based on I, we constructed an ERDF interpretation J s.t. $J \models sk(G)$. Since $sk(G) \models^{ERDF} F$, it follows that $J \models F$. We will show that $I \models F$. We define $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$.

Lemma: For every mapping $u : Var(F) \to Res_J$, it holds $J, u \models F$ iff $I, u \models F$.
Proof: We will prove the Lemma by induction. Without loss of generality, we assume that $\neg$ appears only in front of positive ERDF triples. Otherwise we apply the transformation rules of Definition 5, to get an equivalent formula that satisfies the assumption.

Let $F = p(s, o)$. Assume that $J, u \models F$. Since $V_F \cap sk_G(Var(G)) = \emptyset$, it follows that $p \in V'$, $s, o \in V' \cup Var$, and $J(p) = I(p) \in Prop_I = Prop_J$. Since $\langle [J+u](s), [J+u](o)\rangle \in PT_J(J(p))$, it follows that $\langle [I+u](s), [I+u](o)\rangle \in PT_I(I(p))$. Therefore, $I, u \models F$.

Assume that $I, u \models F$. It follows that $p \in V'$, $s, o \in V' \cup Var$, and $J(p) = I(p) \in Prop_I = Prop_J$. Since $\langle [I+u](s), [I+u](o)\rangle \in PT_I(I(p))$, it follows that $\langle [J+u](s), [J+u](o)\rangle \in PT_J(J(p))$. Therefore, $J, u \models F$.

Let $F = \neg p(s, o)$. Similarly, we prove that $J, u \models F$ iff $I, u \models F$.
Assumption: Assume that the lemma holds for the subformulas of F.
We will show that the lemma holds also for F.

Let $F = \sim G$. It holds: $I, u \models F$ iff $V_G \subseteq V'$ and $I, u \not\models G$ iff $V_G \subseteq V'$ and $J, u \not\models G$ iff $J, u \models F$.

Let $F = F_1 \wedge F_2$. It holds: $I, u \models F$ iff $I, u \models F_1$ and $I, u \models F_2$ iff $J, u \models F_1$ and $J, u \models F_2$ iff $J, u \models F$.

Let $F = \exists x\, G$. It holds: $I, u \models F$ iff $I, u \models \exists x\, G$ iff there is $v : Var(G) \to Res_I$ s.t. $v(y) = u(y)$, $\forall y \in Var(G) - \{x\}$ and $I, v \models G$ iff there is $v : Var(G) \to Res_J$ s.t. $v(y) = u(y)$, $\forall y \in Var(G) - \{x\}$ and $J, v \models G$ iff $J, u \models \exists x\, G$ iff $J, u \models F$.

Let $F = F_1 \vee F_2$ or $F = F_1 \supset F_2$ or $F = \forall x G$. We can prove, similarly to the above cases, that $I, u \models F$ iff $J, u \models F$.
End of lemma

Since $J \models F$, it follows that for every mapping $u : Var(F) \to Res_J$, $J, u \models F$. Therefore, it follows from Lemma that for every mapping $u : Var(F) \to Res_J$, $I, u \models F$. Since $Res_J = Res_I$, it follows that $I \models F$. $\Box$

Proposition 11. Let $O = \langle G, P\rangle$ be an ERDF ontology and let $I, J \in \mathcal{I}^H(O)$ such that $I \leq J$. Then, $Cls_I \subseteq Cls_J$, and for all $c \in Cls_I$, it holds $CT_I(c) \subseteq CT_J(c)$ and $CF_I(c) \subseteq CF_J(c)$.
Proof: Let $c \in Cls_I$. Then, $\langle c, I(Class)\rangle \in PT_I(I(type))$. Note that $J(Class) = I(Class)$ and $J(type) = I(type)$. Thus, $\langle c, J(Class)\rangle \in PT_J(J(type))$, which implies that $c \in Cls_J$.

Let $x \in Cls_I$ and $x \in CT_I(c)$. Then, $\langle x, c\rangle \in PT_I(I(type))$. Thus, $\langle x, c\rangle \in PT_J(J(type))$, which implies that $x \in CT_J(c)$.

Let $x \in Cls_I$ and $x \in CF_I(c)$. Then, $\langle x, c\rangle \in PF_I(I(type))$. Thus, $\langle x, c\rangle \in PF_J(J(type))$, which implies that $x \in CF_J(c)$. $\Box$

Proposition 12. Let $O = \langle G, P\rangle$ be an ERDF ontology and let $M \in \mathcal{M}^{st}(O)$. It holds $M \in \mathcal{M}^H(O)$.

Proof: Let $M \in \mathcal{M}^{st}(O)$. Obviously, $M \in \mathcal{I}^H(O)$ and $M \models sk(G)$. We will show that $M \models r$, $\forall r \in P$. Let $r \in P$. Let v be a mapping $v : Var(r) \rightarrow Res_O^H$ s.t. $M, v \models Cond(r)$. It is enough to show that $M, v \models Concl(r)$.

We now define a total function $v' : FVar(r) \rightarrow V_O$ as follows:

$$v'(x) = \begin{cases} v(x) & \text{if } v(x) \text{ is not the xml value of a well-typed XML literal in } V_O \\ t & \text{if } v(x) \text{ is the xml value of a well-typed XML literal } t \text{ in } V_O \end{cases}$$

Let $x \in V_O$, we define $x^{v'} = x$. Let $x \in FVar(r)$, we define $x^{v'} = v'(x)$. Let F be a formula over V_O, we define $F^{v'}$ to be the formula that results from F after replacing each free variable of F by $v'(x)$. It is easy to see that it holds:
$Concl(r)^{v'} \leftarrow Concl(r)^{v'} \in [r]_{V_O} \subseteq [P]_{V_O}$.

Lemma: Let F be a formula over V_O such that $FVar(F) \subseteq FVar(r)$. Let u be a total function $u : Var(F) \rightarrow Res_O^H$ s.t. $u(x) = v(x)$, $\forall x \in FVar(F)$. It holds: $M, u \models F$ iff $M, u \models F^{v'}$.

Proof: We prove the lemma by induction. Without loss of generality, we assume that $\neg$ appears only in front of positive ERDF triples. Otherwise we apply the transformation rules of Definition 5, to get an equivalent formula that satisfies the assumption.

Let $F = p(s, o)$. It holds: $M, u \models F$ iff $M, u \models p(s, o)$ iff $\langle [M + u](s), [M + u](o) \rangle \in PT_M(M(p))$ iff $\langle [M + u](s^{v'}), [M + u](o^{v'}) \rangle \in PT_M(M(p))$ iff $M, u \models p(s, o)^{v'}$.

Let $F = \neg p(s, o)$. $M, u \models F$ iff $M, u \models p(s, o)$ iff $\langle [M + u](s), [M + u](o) \rangle \in PF_M(M(p))$ iff $\langle [M + u](s^{v'}), [M + u](o^{v'}) \rangle \in PF_M(M(p))$ iff $M, u \models (\neg p(s, o))^{v'}$.

Assumption: Assume that the lemma holds for the subformulas of F.

We will show that the lemma holds also for F.

Let $F = \sim G$. It holds: $M, u \models F$ iff $M, u \models \sim G$ iff $M, u \not\models G$ iff $M, u \not\models G^{v'}$ iff $M, u \models \sim G^{v'}$ iff $M, u \models F^{v'}$.

Let $F = F_1 \wedge F_2$. It holds: $M, u \models F$ iff $M, u \models F_1 \wedge F_2$ iff $M, u \models F_1$ and $M, u \models F_2$ iff $M, u \models F_1^{v'}$ and $M, u \models F_2^{v'}$ iff $M, u \models (F_1 \wedge F_2)^{v'}$ iff $M, u \models F^{v'}$.

Let $F = \exists x G$. It holds: $M, u \models F$ iff $M, u \models \exists x G$ iff there exists $u' : Var(G) \rightarrow Res_O^H$ s.t. $u'(y) = u(y)$, $\forall y \in Var(G) - \{x\}$ s.t. $M, u' \models G$ iff there exists $u' : Var(G) \rightarrow Res_O^H$ s.t. $u'(y) = u(y)$, $\forall y \in Var(G) - \{x\}$ s.t. $M, u' \models G^{v'}$ iff $M, u \models \exists x G^{v'}$ iff $M, u \models F^{v'}$.

Let $F = F_1 \vee F_2$ or $F = F_1 \supset F_2$ or $F = \forall x G$. We can prove, similarly to the above cases, that $M, u \models F$ iff $M, u \models F^{v'}$.

End of Lemma

Since the formula $Cond(r)$ and the mapping v satisfy the conditions of the Lemma ($v(x) = v(x)$, $\forall x \in FVar(Cond(r))$) and $M, v \models Cond(r)$, it follows that $M, v \models Cond(r)^{v'}$. Now since $FVar(Cond(r)^{v'}) = \emptyset$, it follows from Proposition 1 that $M \models Cond(r)^{v'}$. Since $M \in \mathcal{M}^{st}(O)$, it follows that $M \models Concl(r)^{v'}$. Now since $FVar(Concl(r)^{v'}) = \emptyset$, it follows from Proposition 1 that $M, v \models Concl(r)^{v'}$. Further, since the formula $Concl(r)$ and the mapping v satisfy the conditions of the Lemma, it follows that $M, v \models Concl(r)$.

Therefore, $M \models r$, $\forall r \in P$. $\square$

Proposition 13. Let $O = \langle G, P \rangle$ be an ERDF ontology, such that $rdfs{:}subclass(rdf{:}Property, erdf{:}TotalProperty) \in G$. Then, $\mathcal{M}^{st}(O) = \mathcal{M}^H(O)$.

Proof: ¿From Proposition 12, it follows that $\mathcal{M}^{st}(O) \subseteq \mathcal{M}^H(O)$. We will show that $\mathcal{M}^H(O) \subseteq \mathcal{M}^{st}(O)$. Let $M \in \mathcal{M}^H(O)$. It follows that $M \models sk(G)$. We will show that $M \in minimal(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})$.

Let $J \in I^H(O)$ s.t. $J \models sk(G)$ and $J \leq M$. We will show that $J = M$. Since $J \leq M$, it follows that $Prop_J \subseteq Prop_M$ and for all $p \in Prop_J$, it holds $PT_J(p) \subseteq PT_M(p)$ and $PF_J(p) \subseteq PF_M(p)$. Let $p \in Prop_J$. Since $J \models sk(G)$, it follows that

$Prop_J \subseteq TProp_J$. Thus, $p \in TProp_J$. Assume that $PT_J(p) \neq PT_M(p)$. Then, there is $\langle x, y\rangle \in PT_M(p)$ s.t. $\langle x, y\rangle \notin PT_J(p)$. Then, $\langle x, y\rangle \in PF_J(p)$. Thus, $\langle x, y\rangle \in PF_M(p)$, which is impossible, since $\langle x, y\rangle \in PT_M(p)$. Thus, $PT_J(p) = PT_M(p)$. Similarly, we can prove that $PF_J(p) = PF_M(p)$. Therefore, for all $p \in Prop_J$, it holds $PT_J(p) = PT_M(p)$ and $PF_J(p) = PF_M(p)$. We will now show that $Prop_J = Prop_M$. It holds $Prop_J = \{x \in Res^H_O \mid \langle x, Property\rangle \in PT_J(type)\} = \{x \in Res^H_O \mid \langle x, Property\rangle \in PT_I(type)\} = Prop_M$. Based on these results and the fact that $J, M \in \mathcal{I}^H(O)$, it follows that $J = M$. Therefore, $M \in minimal(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})$.

We will now show that $M \in minimal\{I \in \mathcal{I}^H(O) : I \geq M$ and $I \models Concl(r)$, for all $r \in P_{[M,M]}\}$. Since $M \in \mathcal{M}^H(O)$ it follows that $M \in \{I \in \mathcal{I}^H(O) : I \geq M$ and $I \models Concl(r)$, for all $r \in P_{[M,M]}\}$. Let $J \in \{I \in \mathcal{I}^H(O) : I \geq M$ and $I \models Concl(r)$, for all $r \in P_{[M,M]}\}$ and $J \leq M$. Since $J \geq M$, it follows that $Prop_M \subseteq Prop_J$, and for all $p \in Prop_M$, it holds $PT_M(p) \subseteq PT_J(p)$ and $PF_M(p) \subseteq PF_J(p)$. Since $J \leq M$, it follows that $Prop_J \subseteq Prop_M$, and for all $p \in Prop_J$, it holds $PT_J(p) \subseteq PT_M(p)$ and $PF_J(p) \subseteq PF_M(p)$. Therefore, it follows that $Prop_M = Prop_J$, and for all $p \in Prop_M$, it holds $PT_M(p) = PT_J(p)$ and $PF_M(p) = PF_J(p)$. Based on this result and the fact that $J, M \in \mathcal{I}^H(O)$, it follows that $J = M$.

Thus, $M \in minimal\{I \in \mathcal{I}^H(O) : I \geq M$ and $I \models Concl(r)$, for all $r \in P_{[M,M]}\}$.

Since M satisfies the conditions of Definition 21 (Stable Model), it follows that $M \in \mathcal{M}^{st}(O)$.

Thus, it holds $\mathcal{M}^H(O) \subseteq \mathcal{M}^{st}(O)$. □

Proposition 14. Let $O = \langle G, P\rangle$ be an ERDF ontology, and let F, F' be ERDF formulas. If $O \models^{st} F$ and $F \models^{ERDF} F'$ then $O \models^{st} F'$.

Proof:

Let $I \in M^{st}(O)$. Then I is an ERDF interpretation. Since $O \models^{st} F$, it follows that $I \models F$. Since $F \models^{ERDF} F'$, it follows that $I \models F'$. Therefore, $O \models^{st} F'$. □

Proposition 15. Let G, G' be ERDF graphs and F be an ERDF formula.

It holds:

1. If $\langle G, \emptyset\rangle \models^{st} G'$ then $sk(G) \models^{ERDF} G'$.
2. If $sk(G) \models^{ERDF} F$ then $\langle G, \emptyset\rangle \models^{st} F$.

Proof:

1) Let $< G, \emptyset > \models^{st} G'$. We will show that $sk(G) \models^{ERDF} G'$.

Let I be an ERDF interpretation over a vocabulary V s.t. $I \models sk(G)$, we will show that $I \models G'$. We define $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$.

Let $O = \langle G, \emptyset\rangle$. Based on I, we construct a partial interpretation J of V_O as follows:

- $Res_J = Res^H_O$.
- $J_V(x) = x$, for all $x \in V_O \cap URI$.
- We define the mapping: $IL_J : V_O \cap \mathcal{TL} \to Res_J$ such that: $IL_J(x) = x$, if x is a typed literal in V_O other than a well-typed XML literal, and $IL_I(x)$ is the XML value of x, if x is a well-typed XML literal in V_O.
- We define the mapping: $J : V_O \to Res_J$ such that:
  - $J(x) = J_V(x), \ \forall x \in V_O \cap URI$.
  - $J(x) = x, \ \forall\, x \in V_O \cap \mathcal{PL}$.
  - $J(x) = IL_J(x), \ \forall\, x \in V_O \cap \mathcal{TL}$.
- $Prop_J = \{x \in Res_J \mid \exists x' \in V_O,\ J(x') = x$ and $I(x') \in Prop_I\}\}$.
- The mapping $PT_J : Prop_J \to \mathcal{P}(Res_J \times Res_J)$ is defined as follows: $\forall x, y, z \in V_O$, it holds: $\langle J(x), J(y)\rangle \in PT_J(J(z))$ iff $\langle I(x), I(y)\rangle \in PT_I(I(z))$.

– We define the mapping $PF_J : Prop_J \to \mathcal{P}(Res_J \times Res_J)$ as follows:
$\forall x, y, z \in V_O$, it holds:
$\langle J(x), J(y)\rangle \in PF_J(J(z))$ iff $\langle I(x), I(y)\rangle \in PF_I(I(z))$.
– $LV_J = \{x \in Res_J \mid \langle x, J(Literal)\rangle \in PT_J(J(type))\}$.

To show that J is a partial interpretation, it is enough to show that $V_O \cap \mathcal{PL} \subseteq LV_J$. Let $x \in V_O \cap \mathcal{PL}$. Then, $x \in LV_I$. Thus, $\langle x, I(Literal)\rangle \in PT_I(I(type))$. This implies that $\langle x, J(Literal)\rangle \in PT_J(J(type))$. Thus, $x \in LV_J$.

Now, we extend J with the ontological categories:
$Cls_J = \{x \in Res_J \mid \langle x, J(Class)\rangle \in PT_J(J(type))\}$,
$TCls_J = \{x \in Res_J \mid \langle x, J(TotalClass)\rangle \in PT_J(J(type))\}$, and
$TProp_J = \{x \in Res_J \mid \langle x, J(TotalProperty)\rangle \in PT_J(J(type))\}$.
We define the mappings $CT_J, CF_J : Cls_J \to \mathcal{P}(Res_J)$ as follows:
$x \in CT_J(y)$ iff $\langle x, y\rangle \in PT_J(J(type))$, and
$x \in CF_J(y)$ iff $\langle x, y\rangle \in PF_J(J(type))$.

We will now show that J is an ERDF interpretation of V_O. First, we will show that J satisfies semantic condition 2 of Definition 8 (ERDF Interpretation), in a number of steps:
Step 1: Here, we prove that $Res_J = CT_J(J(Resource))$. Obviously, $CT_J(J(Resource)) \subseteq Res_J$. We will show that $Res_J \subseteq CT_J(J(Resource))$. Let $x \in Res_J$. Then, there is $x' \in V_O$ such that $J(x') = x$. We want to show that $\langle J(x'), J(Resource)\rangle \in PT_J(J(type))$. It holds: $\langle J(x'), J(Resource)\rangle \in PT_J(J(type))$ iff $\langle I(x'), I(Resource)\rangle \in PT_I(I(type))$, which is true, since I is an ERDF interpretation that satisfies $sk(G)$ and $I(x') \in Res_I$. Thus, $x = J(x') \in CT_J(J(Resource))$.
Therefore, $Res_J = CT_J(J(Resource))$.

Step 2: Here, we prove that $Prop_J = CT_J(J(Property))$. We will show that $Prop_J \subseteq CT_J(J(Property))$. Let $x \in Prop_J$. Then, there is $x' \in V_O$ such that $J(x') = x$ and $I(x') \in Prop_I$. We want to show that $\langle J(x'), J(Property)\rangle \in PT_J(J(type))$. It holds: $\langle J(x'), J(Property)\rangle \in PT_J(J(type))$ iff $\langle I(x'), I(Property)\rangle \in PT_I(I(type))$, which is true, since $I(x') \in Prop_I$. Thus, $x = J(x') \in CT_J(J(Property))$.
Therefore, $Prop_J \subseteq CT_J(J(Property))$.

We will now show that $CT_J(J(Property)) \subseteq Prop_J$. Let $x \in CT_J(J(Property))$. Then, there is $x' \in V_O$ such that $J(x') = x$. It holds $\langle J(x'), J(Property)\rangle \in PT_J(J(type))$, which implies that $\langle I(x'), I(Property)\rangle \in PT_I(I(type))$. Thus, $I(x') \in Prop_I$ and $x \in Prop_J$.
Therefore, $CT_J(J(Property)) \subseteq Prop_J$.

Step 3: By definition, it holds $Cls_J = CT_J(J(Class))$, $LV_J = CT_J(J(Literal))$, $TCls_J = CT_J(J(TotalClass))$ and $TProp_J = CT_J(J(TotalProperty))$.

We will now show that J satisfies semantic condition 3 of Definition 8 (ERDF Interpretation). Let $\langle x, y\rangle \in PT_J(J(domain))$ and $\langle z, w\rangle \in PT_J(x)$. We will show that $z \in CT_J(y)$. There are $x', y' \in V_O$ such that $J(x') = x$, $J(y') = y$. Thus, $\langle J(x'), J(y')\rangle \in PT_J(J(domain))$. Additionally, there are $z', w' \in V_O$ such that $J(z') = z$, $J(w') = w$. Thus, $\langle J(z'), J(w')\rangle \in PT_J(J(x'))$. Then, $\langle I(x'), I(y')\rangle \in PT_I(I(domain))$ and $\langle I(z'), I(w')\rangle \in PT_I(I(x'))$. Since I is an ERDF interpretation interpretation, $\langle I(z'), I(y'))\rangle \in PT_I(I(type))$. Thus, $\langle J(z'), J(y')\rangle \in PT_J(J(type))$ and $z \in CT_J(y)$.

In a similar manner, we can prove that J also satisfies the rest of the semantic conditions of Definition 8. Thus, J is an ERDF interpretation of V_O.

Moreover, we will show that J is a coherent ERDF interpretation (Definition 9). Assume that this is not the case. Thus, there is $z \in Prop_J$ s.t. $PT_J(z) \cap PF_J(z) \neq \emptyset$.

Thus, there are $x, y \in Res_J$ s.t. $\langle x, y\rangle \in PT_J(z) \cap PF_J(z)$, for such a z. Then, there are $x', y', z' \in V_O$ s.t. $J(x') = x$, $J(y') = y$, and $J(z') = z$. It holds: $\langle J(x'), J(y')\rangle \in PT_J(J(z'))$ and $\langle J(x'), J(y')\rangle \in PF_J(J(z'))$. Thus, $\langle I(x'), I(y')\rangle \in PT_I(I(z'))$ and $\langle I(x'), I(y')\rangle \in PF_I(I(z'))$. But this is impossible, since I is a (coherent) ERDF interpretation. Therefore, J is also a coherent ERDF interpretation.

Thus, $J \in \mathcal{I}^H(O)$.

We will now show that $J \models sk(G)$. Let $p(s, o) \in sk(G)$. It holds $p, s, o \in V_O$. Since $I \models sk(G)$, it holds $I(p) \in Prop_I$. Thus, $\langle I(p), I(Property)\rangle \in PT_I(I(type))$, which implies that $\langle J(p), J(Property)\rangle \in PT_J(J(type))$. ¿From this, it follows that $J(p) \in Prop_J$. It holds: $\langle J(s), J(o)\rangle \in PT_J(J(p))$ iff $\langle I(s), I(o)\rangle \in PT_I(I(p))$. The last statement is true $I \models sk(G)$. Thus, $J \models sk(G)$.

¿From Definition 21 (Stable Model) and the fact that $J \models sk(G)$, it follows that $\exists K \in \mathcal{M}^{st}(O)$ s.t. $K \leq J$. Now from this and the fact that $O \models^{st} G'$, it follows that $K \models G'$. Thus, there is $u : Var(G') \rightarrow Res_O^H$ s.t. $K, u \models G'$.

We will show that $J, u \models G'$.

Let $p(s, o) \in G'$. Since K is an ERDF interpretation of V_O, $K, u \models G'$, and $Prop_K \subseteq Prop_J$, it follows that $p \in V_O$, $s, o \in V_O \cup Var$, and $J(p) = K(p) \in Prop_K \subseteq Prop_J$. Additionally, $\langle [K + u](s), [K + u](o)\rangle \in PT_K(p)$. Since, $\langle [J + u](s), [J + u](o)\rangle = \langle [K + u](s), [K+u](o)\rangle$ and $PT_K(p) \subseteq PT_J(p)$, it follows that $\langle [J+u](s), [J+u](o)\rangle \in PT_J(p)$. Thus, $J, u \models p(s, o)$.

Let $\neg p(s, o) \in G'$. Since K is an ERDF interpretation of V_O, $K, u \models G'$, and $Prop_K \subseteq Prop_J$, it follows that $p \in V_O$, $s, o \in V_O \cup Var$, and $J(p) = K(p) \in Prop_K \subseteq Prop_J$. Additionally, $\langle [K + u](s), [K + u](o)\rangle \in PF_K(p)$. Since, $\langle [J + u](s), [J + u](o)\rangle = \langle [K + u](s), [K+u](o)\rangle$ and $PF_K(p) \subseteq PF_J(p)$, it follows that $\langle [J+u](s), [J+u](o)\rangle \in PF_J(p)$. Thus, $J, u \models \neg p(s, o)$.

We now define a total function $u' : V_{G'} \cup Var(G') \rightarrow V_O$ as follows:

$$u'(x) = \begin{cases} u(x) & \text{if } x \in Var(G'), \text{ and} \\ & \quad u(x) \text{ is not the xml value of a well-typed XML literal in } V_O \\ t & \text{if } x \in Var(G') \text{ and} \\ & u(x) \text{ is the xml value of a well-typed XML literal } t \text{ in } V_O \\ x & \text{otherwise} \end{cases}$$

Moreover, we define a total function $u'' : Var(G') \rightarrow Res_I$ s.t. $u''(x) = I(u'(x))$.

We will show that $I, u'' \models G'$.

Let $p(s, o) \in G'$. Then, $p \in V_{G'}$ and $s, o \in V_{G'} \cup Var$. Since $J \models G'$, it follows that $V_{G'} \subseteq V_O$. Therefore, $V_{G'} \subseteq V_G \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF} \subseteq V'$. Thus, $p \in V'$ and $s, o \in V' \cup Var$.

We will now show that $I(p) \in Prop_I$. It holds:
$\langle I(p), I(Property)\rangle \in PT_I(I(type))$ iff
$\langle J(p), J(Property)\rangle \in PT_J(J(type))$, which holds since $J, u \models G'$.

We want to show that $\langle [I + v''](s), [I + v''](o)\rangle \in PT_I(I(p))$. Note that $\forall x \in V_{G'}$, it holds: $[I + u''](x) = I(u'(x))$ and $J(u'(x)) = [J + u](x)$ (recall the definition of $J(.)$). Moreover, $\forall x \in Var(G')$, it holds: $[I + u''](x) = I(u'(x))$ and $J(u'(x)) = [J + u](x)$. Therefore, it holds:
$\langle [I + u''](s), [I + u''](o)\rangle \in PT_I(I(p))$ iff
$\langle I(u'(s)), I(u'(o))\rangle \in PT_I(I(p)$ iff
$\langle J(u'(s)), J(u'(o))\rangle \in PT_J(J(p)$ iff
$\langle [J+u](s), [J+u](o)\rangle \in PT_J(J(p))$, which is true since $J, u \models G'$. Thus, $I, u'' \models p(s, o)$.

Let $\neg p(s, o) \in G'$. We can show that $I, u'' \models \neg p(s, o)$, in a similar manner.

Thus, $I, u'' \models G'$, which implies that $I \models G'$.

2) Let $sk(G) \models^{ERDF} F$. We will show that $\langle G, \emptyset \rangle \models^{st} F$. In particular, let $O = \langle G, \emptyset \rangle$ and let $I \in \mathcal{M}^{st}(O)$. Note that I is an ERDF interpretation of V_O, such that $I \models sk(G)$. Since $sk(G) \models^{ERDF} F$, it follows that $I \models F$. $\Box$

Proposition 16. Let G, G' be RDF graphs such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$, $V_{G'} \cap \mathcal{V}_{ERDF} = \emptyset$, and $V_{G'} \cap sk_G(Var(G)) = \emptyset$. It holds: $G \models^{RDFS} G'$ iff $\langle G, \emptyset \rangle \models^{st} G'$.
Proof: It follows from Proposition 7 that: $G \models^{RDFS} G'$ iff $G \models^{ERDF} G'$. It follows from Propositions 2 and 10 that: $G \models^{ERDF} G'$ iff $sk(G) \models^{ERDF} G'$. It follows from Propositions 2 and 15 that: $sk(G) \models^{ERDF} G'$ iff $\langle G, \emptyset \rangle \models^{ERDF} G'$.

Therefore, $G \models^{RDFS} G'$ iff $\langle G, \emptyset \rangle \models^{st} G'$. $\Box$

References

1. J. J. Alferes, C. V. Damásio, and L. M. Pereira. A logic programming system for non-monotonic reasoning. *Special Issue of the Journal of Automated Reasoning*, 14(1):93–147, 1995.
2. J. J. Alferes, C. V. Damásio, and L. M. Pereira. Semantic Web Logic Programming Tools. In *International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR'03)*, pages 16–32, 2003.
3. A. Analyti, G. Antoniou, C. V. Damasio, and G. Wagner. Negation and Negative Information in the W3C Resource Description Framework. *Annals of Mathematics, Computing & Teleinformatics (AMCT)*, 1(2):25–34, 2004.
4. G. Antoniou, A. Bikakis, and G. Wagner. A System for Nonmonotonic Rules on the Web. In *3rd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RULEML'03)*, pages 23–36, 2004.
5. G. Antoniou, D. Billington, G. Governatori, and M. J. Maher. Representation results for defeasible logic. *ACM Transanctions on Computational Logic (TOCL)*, 2(2):255–287, 2001.
6. N. Bassiliades, G. Antoniou, and I. P. Vlahavas. DR-DEVICE: A Defeasible Logic System for the Semantic Web. In *2nd International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR'04)*, pages 134–148, 2004.
7. Tim Berners-Lee. Design issues - architectual and philosophical points. Personal notes, 1998. Available at `http://www.w3.org/DesignIssues/`.
8. Carlos Viegas Damásio. SEW - A SEmantic Web engine, 2005. Available at `http://centria.di.fct.unl.pt/~cd/projectos/w4`.
9. Carlos Viegas Damásio and Luís Moniz Pereira. A survey of paraconsistent semantics for logic programas. In D. Gabbay and P. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, Reasoning with Actual and Potential Contradictions. Coordenado por P. Besnard e A. Hunter, pages 241–320. Kluwer Academic Publishers, 1998.
10. F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.
11. F. M. Donini, D. Nardi, and R. Rosati. Description Logics of Minimal Knowledge and Negation as Failure. *ACM Transactions on Computational Logic*, 3(2):177–225, 2002.
12. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in Description Logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, chapter 1, pages 191–236. CSLI Publications, 1996.
13. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. In *9th International Conference on Principles of Knowledge Representation and Reasoning (KR'04)*, pages 141–151, 2004.

14. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-Founded Semantics for Description Logic Programs in the Semantic Web. In *3rd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML'04)*, pages 81–97, 2004.
15. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
16. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. A. Bowen, editors, *5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.
17. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th International Conference on Logic Programming*, pages 579–597. MIT Press, 1990.
18. Patrick Hayes. RDF Semantics. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`.
19. H. Herre, J. Jaspars, and G. Wagner. Partial Logics with Two Kinds of Negation as a Foundation of Knowledge-Based Reasoning. In D.M. Gabbay and H. Wansing, editors, *What Is Negation?* Oxford University Press, 1999.
20. H. Herre and G. Wagner. Stable Models are Generated by a Stable Chain. *Journal of Logic Programming*, 30(2):165–177, 1997.
21. I. Horrocks and P. F. Patel-Schneider. A Proposal for an OWL Rules Language. In *13th International Conference on World Wide Web (WWW'04)*, pages 723–731. ACM Press, 2004.
22. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission, 21 May 2004. Available at `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`.
23. M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, 1995.
24. G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.
25. R. Kowalski and F. Sadri. Logic programs with exceptions. In Warren and Szeredi, editors, *7th International Conference on Logic Programming*. MIT Press, 1990.
26. A. Y. Levy and M. Rousset. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.
27. J. W. Lloyd and R. W. Topor. Making Prolog more Expressive. *Journal of Logic Programming*, 1(3):225–240, 1984.
28. M. J. Maher. A Model-Theoretic Semantics for Defeasible Logic. In *ICLP 2002 workshop on Paraconsistent Computational Logic (PCL 2002)*, pages 255–287, 2002.
29. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/2004/REC-owl-features-20040210/`.
30. B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. In *3rd International Semantic Web Conference (ISWC2004)*, pages 549–563, 2004.
31. L. M. Pereira and J. J. Alferes. Well founded semantics for logic programs with explicit negation. In B. Neumann, editor, *European Conference on Artificial Intelligence*, pages 102–106. John Wiley & Sons, 1992.
32. P. Rao, K. F. Sagonas, T. Swift, D. S. Warren, and J. Freire. XSB: A System for Efficiently Computing WFS. In *Proceedings of 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'97)*, pages 1070–1080, July 1997.
33. Riccardo Rosati. Towards expressive KR systems integrating Datalog and Description Logics: Preliminary report. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 160–164, 1999.

34. The rule markup initiative (ruleml). Available at `http://www.ruleml.org`.
35. C. Sakama and K. Inoue. Paraconsistent Stable Semantics for extended disjunctive programs. *Journal of Logic and Computation*, 5(3):265–285, 1995.
36. M. Sintek and S. Decker. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *First International Semantic Web Conference on The Semantic Web (ISWC2002)*, pages 364–378. Springer-Verlag, 2002.
37. H. J. ter Horst. Extending the RDFS Entailment Lemma. In *3rd International Semantic Web Conference (ISWC2004)*, pages 77–91, 2004.
38. Tim-Berners-Lee. Notation 3 - An RDF language for the Semantic Web. W3C Recommendation, 1998. Available at `http://www.w3.org/DesignIssues/Notation3.html`.
39. G. Wagner. A Database Needs Two Kinds of Negation. In *3rd Symposium on Mathematical Fundamentals of Database and Knowledge Base Systems (MFDBS'91)*, pages 357–371. Springer-Verlag, 1991.
40. G. Wagner. Web Rules Need Two Kinds of Negation. In *1st International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR'03)*. Springer-Verlag, December 2003.
41. G. Yang, M. Kifer, and C. Zhao. Flora-2: A Rule-Based Knowledge Representation and Inference Infrastructure for the Semantic Web. In *2nd International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE'03)*, pages 671–688, 2003.
42. Guizhen Yang and Michael Kifer. Inheritance and Rules in Object-Oriented Semantic Web Languages. In *2nd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RULEML'03)*, pages 95–110, 2003.
43. Guizhen Yang and Michael Kifer. Reasoning about Anonymous Resources and Meta Statements on the Semantic Web. *Journal on Data Semantics*, 1:69–97, 2003.