Journal of Artificial Intelligence Research 32 (2008) 37-94 Submitted 08/07; published 05/08

Extended RDF as a Semantic Foundation of Rule Markup Languages

Anastasia Analyti ANALYTI@ICS.FORTH.GR
Institute of Computer Science, FORTH-ICS, Crete, Greece

Grigoris Antoniou ANTONIOU@ICS.FORTH.GR
Institute of Computer Science, FORTH-ICS, Crete, Greece
Department of Computer Science, University of Crete, Greece

Carlos Viegas Damásio CD@DI.FCT.UNL.PT
Centro de Inteligência Artificial, Universidade Nova de Lisboa,
Caparica, Portugal

Gerd Wagner G.WAGNER@TU-COTTBUS.DE
Institute of Informatics, Brandenburg University
of Technology at Cottbus, Germany

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.

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. Berners-Lee (1998) identifies the following fundamental theoretical problems: 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 (McGuinness & van Harmelen, 2004), which is based on Description Logics (DLs) (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003), includes a form of classical negation through class complements, this form is limited. This

©2008 AI Access Foundation. All rights reserved.

is because, to achieve decidability, classes are formed based on specific class constructors and negation on properties is not fully 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, demonstrated by the Rule Markup Initiative[1], has restarted the discussion of the fundamentals of closed-world reasoning and the appropriate mechanisms to implement it in rule systems.

The RDF(S)[2] recommendation (Klyne & Carroll, 2004; Hayes, 2004) 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 representations of strings, numbers, dates, or other basic datatypes.

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)[3], 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.

However, RDF(S) does not support negation and rules. Wagner (1991) argues 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 a subsequent paper, Wagner (2003) makes also this point for the Semantic Web, as a framework for knowledge representation in general. In the present paper, we make the same argument for the Semantic Web language RDF and show how it can be extended to accommodate the two negations of Partial Logic (Herre, Jaspars, & Wagner, 1999), as well as derivation rules. We call the new language *Extended RDF* and denote it by *ERDF*.

1. http://www.ruleml.org/
2. RDF(S) stands for *Resource Description Framework (Schema)*.
3. http://gbiv.com/protocols/uri/rfc/rfc3986.html

The model-theoretic semantics of ERDF, called *ERDF stable model semantics*, is developed based on Partial Logic (Herre et al., 1999).

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 $\mathit{rdf}{:}\mathit{type}(x,c)$.

Partial logic also allows to distinguish between properties (and classes) that are completely represented in a knowledge base and those that are not. The classification if a property is completely represented or not is up to the owner of the knowledge base: the owner must know for which properties there is complete information and for which there is not. Clearly, in the case of a completely represented (*closed*) property p, entailment of $\sim p(x,y)$ allows to derive $\neg p(x,y)$, and the underlying *completeness assumption* has also been called *Closed-World Assumption (CWA)* in the AI literature.

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 the rule $\neg \mathit{rdf}{:}\mathit{type}(?x,c) \leftarrow \sim \mathit{rdf}{:}\mathit{type}(?x,c)$. These derivation rules are called *default closure rules*. In the case of a total property p, default closure rules are not applicable. This is because, some of the considered interpretations will satisfy $p(x,y)$ and the rest $\neg p(x,y)$[4], preventing the preferential entailment of $\sim p(x,y)$. Thus, on total properties, an *Open-World Assumption (OWA)* applies. Similarly to first-order-logic, in order to infer negated statements about total properties, explicit negative information has to be supplied, along with ordinary (positive) information.

As an example, consider an ERDF knowledge base KB that contains the facts:

$$\mathit{interestedIn}(\mathit{Anastasia}, \mathit{SemanticWeb}) \qquad \mathit{interestedIn}(\mathit{Grigoris}, \mathit{Robotics})$$

indicating that *Anastasia* is interested in the *SemanticWeb* area and *Grigoris* is interested in the *Robotics* area. Then, the statement $\mathit{interestedIn}(\mathit{Anastasia}, \mathit{Robotics})$ is not satisfied in the single intended model of KB. Thus, KB entails $\sim \mathit{interestedIn}(\mathit{Anastasia}, \mathit{Robotics})$.

Assume now that the previous list of areas of interest is not complete for *Anastasia* or *Grigoris*. Then, we should add to knowledge base KB the statement:

$$\mathit{rdf}{:}\mathit{type}(\mathit{interestedIn},\ \mathit{erdf}{:}\mathit{TotalProperty})$$

indicating that *interestedIn* is a total property. In this case, an open-world assumption is made for *interestedIn* and KB does not entail $\sim \mathit{interestedIn}(\mathit{Anastasia}, \mathit{Robotics})$, any longer. In particular, there is an intended model of the revised KB that satisfies $\mathit{interestedIn}(\mathit{Anastasia}, \mathit{Robotics})$. Of course, if it is known that *Anastasia* is not interested in *Robotics* then $\neg \mathit{interestedIn}(\mathit{Anastasia}, \mathit{Robotics})$ should be added to KB.

Assume now that we add to KB the following facts:

4. On total properties p, the *Law of Excluded Middle* $p(x,y) \vee \neg p(x,y)$ applies.

$$hasCar(Anastasia, Suzuki) \qquad hasCar(Grigoris, Volvo)$$

and assume that KB has complete knowledge on the property $hasCar$, as far as it concerns elements in the Herbrand Universe of KB. Then, the default closure rule $\neg hasCar(?x, ?y) \leftarrow \sim hasCar(?x, ?y)$ can be safely added to KB. As a result, $\neg hasCar(Anastasia, Volvo)$ is satisfied in all intended models of KB. Thus, KB entails $\neg hasCar(Anastasia, Volvo)$.

The previous example shows the need for supporting both closed-world and open-world reasoning in the same framework. Damasio et al. (2006) and Analyti et al. (2004) provide further examples and arguments for this need. Unfortunately, classical logic and thus also OWL support only open-world reasoning.

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 the 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. Particularly, we consider only *coherent* ERDF interpretations (imposing coherence on all properties). Thus, in this paper, 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, the Herbrand interpretations, and the minimal Herbrand models of an ERDF ontology. Since not all minimal Herbrand models of an ERDF ontology are intended, we define the *stable models* of an ERDF ontology. The definition of a stable model is based on the intuition that:

   (a) assertions stating that a property p or class c is total should only be accepted, if the ontology contains some direct support for them in the form of an acceptable rule sequence, and

   (b) assertions $[\neg]p(s, o)$ and $[\neg]rdf{:}type(o, c)$ should only be accepted, if (i) the ontology contains some direct support for them in the form of an acceptable rule sequence, or (ii) property p and class c are total, respectively.

6. 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. Moreover, we show that if all properties are total, (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 framework with respect to Partial Logic (Herre et al., 1999) 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 Partial Logic (Herre et al., 1999), 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.

This work extends our conference paper (Analyti, Antoniou, Damásio, & Wagner, 2005) by (i) considering the full RDFS model, (ii) providing a detailed characterization of the properties of ERDF interpretations/models, Herbrand interpretations/models, and finally ERDF stable models, (iii) discussing decidability issues, and (iv) providing formal proofs of all lemmas and propositions.

The rest of the paper is organized as follows: In Section 2, we extend RDF graphs to ERDF graphs and ERDF formulas. Section 3 defines ERDF interpretations and ERDF entailment. We show that ERDF entailment extends RDFS entailment. In Section 4, we define ERDF ontologies and the Herbrand models of an ERDF ontology. In Section 5, we define the stable models of an ERDF ontology. Section 6 defines stable model entailment, showing that it extends ERDF entailment. In Section 7, we provide a brief sketch of the ERDF/XML syntax. Decidability issues for the ERDF stable model semantics are discussed in Section 8. Section 9 shows that the developed ERDF model theory can be seen as a Tarski-style model theory. Section 10 reviews related work and Section 11 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 lemmas and propositions, presented in the paper.

2. 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 (Klyne & Carroll, 2004; Hayes, 2004), *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 an 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/Riesling") 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 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 $\mathcal{URI}$, 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: $\mathcal{URI} \cap \mathcal{LIT} = \emptyset$.

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

An RDF triple (Klyne & Carroll, 2004; Hayes, 2004) is a triple "$s\ p\ o.$", where $s \in \mathcal{URI} \cup Var$, $p \in \mathcal{URI}$, and $o \in \mathcal{URI} \cup \mathcal{LIT} \cup Var$, 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 "$s\ p\ o.$" by $p(s, o)$. Below we extend the notion of RDF triple to allow for both positive and negative information.

Definition 2.1 (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*[5] and *object*, respectively, and $p \in V \cap \mathcal{URI}$ 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. $\Box$

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, $ex{:}denotationOf(\text{"}Grigoris\text{"}, ex{:}Grigoris)$ 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). Prud'hommeaux & Seaborne (2008) and de Bruijn et al. (2005) also consider literals in the subject position of RDF triples.

Based on the notion of ERDF triple, we define ERDF graphs and ERDF formulas, as follows:

Definition 2.2 (ERDF graph) An *ERDF graph* G is a set of ERDF triples over some vocabulary V. We denote the variables appearing in G by $Var(G)$, and the set of URI references and literals appearing in G by V_G. $\Box$

Note that as an RDF graph is a set of RDF triples (Klyne & Carroll, 2004; Hayes, 2004), an RDF graph is also an ERDF graph.

Definition 2.3 (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 $x \in Var$. An *ERDF formula* over V is an element of $L(V)$. We denote the set of variables appearing

5. Opposed to "pure" RDF (Klyne & Carroll, 2004), we allow literals in the subject position of an ERDF triple.

in F by $Var(F)$, and the set of free variables[6] appearing in F by $FVar(F)$. Moreover, we denote the set of URI references and literals appearing in F by V_F. $\square$

For example, let:

$$F = \forall ?x\ \exists ?y\ (rdf{:}type(?x, ex{:}Person) \supset ex{:}hasChild(?y, ?x)) \land\ rdf{:}type(?z, ex{:}Person)$$

Then, F is an ERDF formula over the vocabulary $V = \{rdf{:}type, ex{:}Person, ex{:}hasChild\}$ with $Var(F) = \{?x, ?y, ?z\}$ and $FVar(F) = \{?z\}$.

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.

3. 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, based on the notion of *partial interpretation*.

3.1 Partial Interpretations

We define a partial interpretation as an extension of a simple interpretation (Hayes, 2004), where each property is associated not only with a truth extension but also with a falsity extension allowing for partial properties. The notation $\mathcal{P}(S)$, where S is a set, denotes the *powerset* of S.

Definition 3.1 (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.
- A set of properties $Prop_I$.
- A vocabulary interpretation mapping I_V[7]: $V \cap \mathcal{URI} \to Res_I \cup Prop_I$.
- 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:

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

6. Without loss of generality, we assume that a variable cannot have both free and bound occurrences in F, and more than one bound occurrence.
7. In the symbol I_V, V stands for *Vocabulary*.

Note that the truth and falsity extensions of a property p according to a partial interpretation I, that is $PT_I(p)$ and $PF_I(p)$, are sets of pairs $\langle subject, object\rangle$ of resources. As an example, let:

$$V = \{ex{:}Carlos,\ ex{:}Grigoris,\ ex{:}Riesling,\ ex{:}likes,\ ex{:}denotationOf,\ \text{“}Grigoris\text{”}\hat{}\hat{}xsd{:}string\}$$

and consider a structure I that consists of:

- A set of resources $Res_I = \{C, G, R, l, d, \text{“}Grigoris\text{”}\}$.
- A set of properties $Prop_I = \{l, d\}$.
- A vocabulary interpretation mapping $I_V : V \cap \mathcal{URI} \rightarrow Res_I \cup Prop_I$ such that: $I_V(ex{:}Carlos) = C$, $I_V(ex{:}Grigoris) = G$, $I_V(ex{:}Riesling) = R$, $I_V(ex{:}likes) = l$, and $I_V(ex{:}denotationOf) = d$.
- A property-truth extension mapping $PT_I : Prop_I \rightarrow \mathcal{P}(Res_I \times Res_I)$ such that: $PT_I(d) = \{\langle \text{“}Grigoris\text{”}, G\rangle\}$.
- A property-falsity extension mapping $PF_I : Prop_I \rightarrow \mathcal{P}(Res_I \times Res_I)$ such that: $PF_I(l) = \{\langle C, R\rangle\}$.
- A mapping $IL_I : V \cap \mathcal{TL} \rightarrow Res_I$ such that: $IL_I(\text{“}Grigoris\text{”}\hat{}\hat{}xsd{:}string) = \text{“}Grigoris\text{”}$.
- A set of literal values $LV_I = \{\text{“}Grigoris\text{”}\}$.

It is easy to see that I is a partial interpretation of V, expressing that: (i) “*Grigoris*” is the denotation of *Grigoris* and (ii) *Carlos* dislikes *Riesling*.

Definition 3.2 (Coherent partial interpretation) A partial 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 partial interpretations enforce the constraint that a pair of resources cannot belong to both the truth and falsity extensions of a property (i.e., all properties are coherent). Intuitively, this means that an ERDF triple cannot be both true and false.

Continuing our previous example, note that I is a coherent partial interpretation. Consider now a partial interpretation J which is exactly as I, except that it also holds: $PT_J(l) = \{\langle C, R\rangle\}$ (expressing that *Carlos* likes *Riesling*). Then, $\langle C, R\rangle$ belongs to both the truth and falsity extension of l (i.e., $\langle C, R\rangle \in PT_J(l) \cap PF_J(l)$). Thus, J is not coherent.

To define satisfaction of an ERDF formula w.r.t. a partial interpretation, we need first the following auxiliary definition.

Definition 3.3 (Composition of a partial interpretation and a valuation) Let I be a partial interpretation of a vocabulary V and let v be a partial function $v : Var \rightarrow Res_I$ (called *valuation*). We define: (i) $[I + v](x) = v(x)$, if $x \in Var$, and (ii) $[I + v](x) = I(x)$, if $x \in V$. $\square$

Definition 3.4 (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. Additionally, let v be a mapping $v : Var(F) \rightarrow Res_I$.

- If $F = p(s, o)$ then $I, v \models F$ iff $p \in V \cap \mathcal{URI}$, $s, o \in V \cup Var$, $I(p) \in Prop_I$, 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 $p \in V \cap \mathcal{URI}$, $s, o \in V \cup Var$, $I(p) \in Prop_I$, 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[8] $I, v \models F$ iff $I, v \models {\sim}F_1 \vee F_2$.
- If $F = \exists x\ G$ then $I, v \models F$ iff there exists mapping $u : Var(G) \rightarrow Res_I$ 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 $u : Var(G) \rightarrow Res_I$ 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:
  $\neg(F \wedge G) \rightarrow \neg F \vee \neg G$, $\neg(F \vee G) \rightarrow \neg F \wedge \neg G$, $\neg(\neg F) \rightarrow F$, $\neg({\sim} F) \rightarrow F$[9],
  $\neg(\exists x\ F) \rightarrow \forall x\ \neg F$, $\neg(\forall x\ F) \rightarrow \exists x\ \neg F$, $\neg(F \supset G) \rightarrow F \wedge \neg G$. □

Continuing our previous example, let $v : \{?x, ?y, ?z\} \rightarrow Res_I$ such that $v(?x) = C$, $v(?y) = R$, and $v(?z) = G$. It holds:

$$I, v \models \neg ex{:}likes(?x, ?y) \wedge ex{:}denotationOf(\text{“}Grigoris\text{”\^{}\^{}}xsd{:}string,\ ?z).$$

Definition 3.5 (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 $v : Var(F) \rightarrow Res_I$, it holds $I, v \models F$. □

Continuing our previous example, $I \models \exists ?x\ \neg ex{:}likes(ex{:}Carlos, ?x)$.

Below we define ERDF graph satisfaction, extending satisfaction of an RDF graph (Hayes, 2004) (see also Appendix A).

Definition 3.6 (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 $v : Var(G) \rightarrow Res_I$. We define:

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

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 $Var(G) = \{x_1, ..., x_k\}$. Then, G represents the ERDF formula $formula(G) = \exists ?x_1, ..., \exists ?x_k\ t_1 \wedge ... \wedge t_n$. This is shown in the following lemma.

8. *Material implication* is the logical relationship between any two ERDF formulas such that either the first is *non-true* or the second is *true*.
9. This transformation expresses that if it is *false* that F *does not hold* then F *holds*.

Lemma 3.1 Let G be an ERDF graph and let I be a partial interpretation of a vocabulary V. It holds: $I \models_{\texttt{GRAPH}} G$ iff $I \models \mathit{formula}(G)$.

Following the RDF terminology (Klyne & Carroll, 2004), the variables of an ERDF graph are also called *blank nodes* and intuitively denote anonymous web resources. For example, consider the ERDF graph:

$$G = \{\mathit{rdf}{:}\mathit{type}(?x, \mathit{ex}{:}\mathit{EuropeanCountry}),\ \neg \mathit{rdf}{:}\mathit{type}(?x, \mathit{ex}{:}\mathit{EUmember})\}.$$

Then, G represents the ERDF formula $\mathit{formula}(G) =$

$$\exists ?x\ (\mathit{rdf}{:}\mathit{type}(?x, \mathit{ex}{:}\mathit{EuropeanCountry}) \wedge \neg \mathit{rdf}{:}\mathit{type}(?x, \mathit{ex}{:}\mathit{EUmember})),$$

expressing that there is a European country which is not a European Union member.

Notational Convention: *Let G be an ERDF graph, let I be a partial interpretation of a vocabulary V, and let v be a mapping $v : \mathit{Var}(G) \rightarrow \mathit{Res}_I$. Due to Lemma 3.1, we will write (by an abuse of notation) "$I, v \models G$" and "$I \models G$" instead of "$I, v \models_{\texttt{GRAPH}} G$" and "$I \models_{\texttt{GRAPH}} G$", respectively.*

3.2 ERDF Interpretations and Entailment

In this subsection, we define ERDF interpretations and entailment as an extension of RDFS interpretations and entailment (Hayes, 2004). First, we define the vocabularies of RDF, RDFS, and ERDF.

The vocabulary of RDF, $\mathcal{V}_{RDF}$, is a set of $\mathcal{URI}$ references in the *rdf*: namespace (Hayes, 2004), as shown in Table 1. The vocabulary of RDFS, $\mathcal{V}_{RDFS}$, is a set of $\mathcal{URI}$ references in the *rdfs*: namespace (Hayes, 2004), as shown in Table 1. The *vocabulary of ERDF*, $\mathcal{V}_{ERDF}$, is a set of $\mathcal{URI}$ references in the *erdf*: namespace. Specifically, the set of ERDF predefined classes is $\mathcal{C}_{ERDF} = \{\mathit{erdf}{:}\mathit{TotalClass}, \mathit{erdf}{:}\mathit{TotalProperty}\}$. 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 (Hayes, 2004) (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 3.7 (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 $\mathit{Cls}_I \subseteq \mathit{Res}_I$ for classes, $\mathit{TCls}_I \subseteq \mathit{Cls}_I$ for total classes, and $\mathit{TProp}_I \subseteq \mathit{Prop}_I$ for total properties, as well as the class-truth extension mapping $\mathit{CT}_I : \mathit{Cls}_I \rightarrow \mathcal{P}(\mathit{Res}_I)$, and the class-falsity extension mapping $\mathit{CF}_I : \mathit{Cls}_I \rightarrow \mathcal{P}(\mathit{Res}_I)$, such that:

1. $x \in \mathit{CT}_I(y)$ iff $\langle x, y\rangle \in \mathit{PT}_I(I(\mathit{rdf}{:}\mathit{type}))$, and
   $x \in \mathit{CF}_I(y)$ iff $\langle x, y\rangle \in \mathit{PF}_I(I(\mathit{rdf}{:}\mathit{type}))$.

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:$\_i, \quad \forall i \in \{1, 2, ...\}$	$rdfs$:$label$
rdf:$value$	

Table 1: The vocabulary of RDF and RDFS

2. The ontological categories are defined as follows:
   $Prop_I = CT_I(I(rdf{:}Property))$ $\quad$ $Cls_I = CT_I(I(rdfs{:}Class))$
   $Res_I = CT_I(I(rdfs{:}Resource))$ $\quad$ $LV_I = CT_I(I(rdfs{:}Literal))$
   $TCls_I = CT_I(I(erdf{:}TotalClass))$ $\quad$ $TProp_I = CT_I(I(erdf{:}TotalProperty))$.

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 (Hayes, 2004), shown in Table 2 and Table 3 of Appendix A, respectively.
17. I satisfies the following triples, called *ERDF axiomatic triples*:
    rdfs:*subClassOf*(*erdf*:*TotalClass*, *rdfs*:*Class*).
    rdfs:*subClassOf*(*erdf*:*TotalProperty*, *rdfs*:*Class*).

Note that while RDFS intepretations (Hayes, 2004) imply a two-valued interpretation of the instances of *rdf*:*Property*, this is no longer the case with ERDF interpretations. Specifically, let I be an ERDF interpretation, let $p \in CT_I(I(rdf{:}Property))$, and let $\langle x, y\rangle \in Res_I \times Res_I$. It may be the case that neither $\langle x, y\rangle \in PT_I(p)$ nor $\langle x, y\rangle \in PF_I(p)$. That is $p(x, y)$ is neither true nor false.

Semantic conditions of ERDF interpretations may impose constraints to both the truth and falsity extensions of properties and classes. Specifically, consider semantic condition 6 of Definition 3.7 and assume that $\langle x, y\rangle \in PT_I(I(rdfs{:}subClassOf))$. Then, I should not only satisfy $CT_I(x) \subseteq CT_I(y)$ (as an RDFS interpretation I does), but also $CF_I(y) \subseteq CF_I(x)$. The latter is true because if it is certain that a resource z does not belong to the truth extension of class y then it is certain that z does not belong to the truth extension of class x. Thus, the falsity extension of y is contained in the falsity extension of x. Similar is the case for semantic condition 8. Semantic conditions 12 and 13 represent our definition of total classes and total properties, respectively. Semantic condition 15 expresses that the denotation of an ill-typed XML literal is not a literal value. Therefore (see semantic condition 2), it is certain that it is not contained in the truth extension of the class *rdfs*:*Literal*. Thus, it is contained in the falsity extension of the class *rdfs*:*Literal*.

Let I be a coherent ERDF interpretation of a vocabulary V. Since $I(rdf{:}type) \in Prop_I$, it holds: $\forall x \in Cls_I,\ \ CT_I(x) \cap CF_I(x) = \emptyset$. Thus, all properties and classes of coherent ERDF interpretations are coherent.

Convention: *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. Moreover, to improve the readability of our examples, we will ignore the example namespace ex:.*

According to RDFS semantics (Hayes, 2004), 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[10]. Such a triple is called *XML clash*. To

10. In RDF(S), literals are not allowed in the subject position of RDF triples, whereas blank nodes are. For this reason, before the RDF and RDFS entailment rules are applied on an RDF graph, each literal is replaced by a unique blank node. This way inferences can be drawn on the literal value denoted by this literal, without concern for the above restriction (Hayes, 2004).

understand this, note that from semantic condition 3 of Definition A.3 (RDF interpretation, Appendix A), it follows that the denotation of an ill-typed XML literal cannot be a literal value. Now, from semantic conditions 1 and 2 of Definition A.5 (RDFS interpretation, Appendix A), it follows that the denotation of an ill-typed XML literal cannot be of type *rdfs*:*Literal*. Therefore, the derivation of an XML clash from an RDF graph G through the application of the RDF and RDFS entailment rules, indicates that there is no RDFS interpretation that satisfies G.

An ERDF graph can be ERDF-inconsistent[11], not only due to the appearance of an ill-typed XML literal in the ERDF graph (in combination with semantic condition 15 of Definition 3.7), but also due to the additional semantic conditions for coherent ERDF interpretations.

For example, let $p, q, s, o \in \mathcal{URI}$ 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 3.1 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, $I \models rdf{:}type(x,c) \vee \neg rdf{:}type(x,c)$).

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

Definition 3.8 (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 $I \models F'$. $\square$

For example, let:

$F = \forall ?x\ \exists ?y\ (rdf{:}type(?x, Person) \supset hasFather(?x, ?y))\ \wedge rdf{:}type(John, Person)$.

Additionally, let $F' = \exists ?y\ hasFather(John, ?y)\ \wedge\ rdf{:}type(hasFather, rdf{:}Property)$.
Then $F \models^{ERDF} F'$.

The following proposition shows that ERDF entailment extends RDFS entailment (Hayes, 2004) (see also Appendix A) from RDF graphs to ERDF formulas. In other words, ERDF entailment is upward compatible with RDFS entailment.

Proposition 3.2 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 $G \models^{ERDF} G'$.

It easily follows from Proposition 3.2 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.

11. Meaning that there is no (coherent) ERDF interpretation that satisfies the ERDF graph.

4. ERDF Ontologies & Herbrand Interpretations

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 minimal Herbrand models of an ERDF ontology.

Definition 4.1 (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 $G \in L(V|\{\neg\}) \cup \{false\}$ 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 $FVar(r)$[12], 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. □

Recall that $L(V|\{\neg\})$ denotes the set of ERDF triples over V. Therefore, the conclusion of an ERDF rule, unless it is *false*, it is either a positive ERDF triple $p(s, o)$ or a negative ERDF triple $\neg p(s, o)$.

For example, consider the derivation rule r:

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

Then, r is an ERDF rule, indicating that between two classes P and Q, it holds $allRelated(P, Q)$ if for all instances p of the class P, there is an instance q of the class Q such that it holds $related(p, q)$. Note that $Var(r) = \{?P, ?Q, ?p, ?q\}$ and $FVar(r) = \{?P, ?Q\}$.

When $Cond(r) = true$ and $Var(r) = \{\}$, rule r is called *ERDF fact*. When $Concl(r) = false$, rule r is called *ERDF constraint*. We assume that for every partial interpretation I and every function $v : Var \to Res_I$, it holds $I, v \models true$, $I \models true$, $I, v \not\models false$, and $I \not\models false$.

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 4.2 (ERDF ontology) An *ERDF ontology* (or *ERDF 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 4.3 (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 $v : Var(r) \to Res_I$ 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 $I \models r$, $\forall\, r \in P$. □

12. $FVar(r) = FVar(F) \cup FVar(G)$.

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 4.4 (Skolemization of an ERDF graph) Let G be an ERDF graph. The *skolemization function* of G is an 1:1 mapping $sk_G : Var(G) \rightarrow \mathcal{URI}$, 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 $sk_G(x)$. $\Box$

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: $G = \{rdf{:}type(?x, EuropeanCountry),\ \neg rdf{:}type(?x, EUmember)\}$. Then,

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

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 4.1 Let G be an ERDF graph and let 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 4.5 (Vocabulary of an ERDF ontology) Let $O = \langle G, P \rangle$ be an ERDF ontology. The *vocabulary* of O is defined as $V_O = V_{sk(G)} \cup V_P \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. $\Box$

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 4.6 (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. We say that I is a *Herbrand interpretation* of O iff:

- $Res_I = Res^H_O$.
- $I_V(x) = x$, for all $x \in V_O \cap \mathcal{URI}$.
- $IL_I(x) = x$, 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 $\mathcal{M}^H(O)$. $\Box$

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.

It is easy to see that every Herbrand model of an ERDF ontology O is a model of O. Moreover, note that every Herbrand interpretation of an ERDF ontology O is uniquely identified by (i) its set of properties and (ii) its property-truth and property-falsity extension mappings.

However, not all Herbrand models of an ERDF ontology O are desirable. For example, let $p, s, o \in \mathcal{URI}$, 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. This is because p is not a total property and $p(o, s)$ cannot be derived from O (negation-as-failure)[13] .

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

Definition 4.7 (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 the relation $\leq$ is reflexive, transitive, and antisymmetric. Thus, it is a partial ordering on $\mathcal{I}^H(O)$.

The intuition behind Definition 4.7 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.

The following proposition expresses that two Herbrand interpretations I, J of an ERDF ontology O are incomparable, if the property-truth or property-falsity extension of a total property p w.r.t. I and J are different.

Proposition 4.2 Let $O = \langle G, P\rangle$ be an ERDF ontology and let $I, J \in \mathcal{I}^H(O)$. Let $p \in TProp_I \cap TProp_J$. If $PT_I(p) \neq PT_J(p)$ or $PF_I(p) \neq PF_J(p)$ then $I \not\leq J$ and $J \not\leq I$.

Definition 4.8 (Minimal Herbrand interpretations) Let O be an ERDF ontology and let $\mathcal{I} \subseteq \mathcal{I}^H(O)$. We define $minimal(\mathcal{I}) = \{I \in \mathcal{I} \mid\ \nexists J \in \mathcal{I} :\ J \neq I \text{ and } J \leq I\}$. $\Box$

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

13. On the other hand, if p is a total property then $p(o, s) \vee \neg p(o, s)$ should be satisfied by all intended models. Therefore, in this case, there should be an intended model of O that satisfies $p(o, s)$.

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 non-monotonic 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 the help of an example. Consider the rule $p \leftarrow \sim q$ whose model set, according to the stable model semantics (Gelfond & Lifschitz, 1988, 1990; Herre & Wagner, 1997; Herre et al., 1999), 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 $p, q, s, o \in \mathcal{URI}$. 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 4.9 (Grounding of an ERDF program) Let V be a vocabulary and let r be an ERDF rule. We denote by $[r]_V$ the set of rules that result from r if we replace each variable $x \in FVar(r)$ by $v(x)$, for all mappings $v : FVar(r) \rightarrow V$.

Let P be an ERDF program. We define $[P]_V = \bigcup_{r \in P} [r]_V$. $\Box$

Note that a rule variable can naturally appear in the subject position of an ERDF triple. Since variables can be instantiated by a literal, a literal can naturally appear in the subject position of an ERDF triple in the grounded version of an ERDF program. This case further supports our choice of allowing literals in the subject position of an ERDF triple.

5. ERDF Stable Models

In this section, we define the intended models of an ERDF ontology 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.

Below, we define the stable models of an ERDF ontology, based on the coherent stable models[14] of Partial Logic (Herre et al., 1999).

Definition 5.1 (ERDF 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 an *(ERDF) stable model* of O iff there is a chain of Herbrand interpretations of O, $I_0 \leq ... \leq I_{k+1}$ such that $I_k = I_{k+1} = M$ and:

14. Note that these models on extended logic programs are equivalent (Herre et al., 1999) to Answer Sets of answer set semantics (Gelfond & Lifschitz, 1990).

1. $I_0 \in minimal(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})$.

2. For successor ordinals α with $0 < \alpha \leq k+1$:
$I_\alpha \in minimal(\{I \in \mathcal{I}^H(O) \mid I \geq I_{\alpha-1}$ and $I \models Concl(r),\ \forall r \in P_{[I_{\alpha-1},M]}\})$, where
$P_{[I_{\alpha-1},M]} = \{r \in [P]_{V_O} \mid I \models Cond(r),\ \forall I \in \mathcal{I}^H(O)$ s.t. $I_{\alpha-1} \leq I \leq M\}$.

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

Note that I_0 is a minimal Herbrand interpretation of $O = \langle G, P\rangle$ that satisfies $sk(G)$, while Herbrand interpretations $I_1, ..., I_{k+1}$ correspond to a stratified sequence of rule applications, where all applied rules remain applicable throughout the generation of a stable model M. In our words, a stable model is generated bottom-up by the iterative application of the rules in the ERDF program P, starting from the information in the ERDF graph G. Thus, ERDF stable model semantics, as a refinement of minimal model semantics, captures the intuition that:

- Assertions $rdf{:}type(p, erdf{:}TotalProperty)$ and $rdf{:}type(c, erdf{:}TotalClass)$ should only be accepted if the ontology contains some direct support for them in the form of an acceptable rule sequence (that corresponds to a proof).

- Assertions $p(s, o)$ and $\neg p(s, o)$ should only be accepted if the ontology contains some direct support for them in the form of an acceptable rule sequence, or $rdf{:}type(p, erdf{:}TotalProperty)$ is accepted.

- Assertions $rdf{:}type(o, c)$ and $\neg rdf{:}type(o, c)$ should only be accepted if the ontology contains some direct support for them in the form of an acceptable rule sequence, or $rdf{:}type(c, erdf{:}TotalClass)$ is accepted.

Wine Selection Example: Consider a class *Wine* whose instances are wines, and a property $likes(X, Y)$ indicating that person X likes object Y. Assume now that we want to select wines for a dinner such that, for each guest, there is on the table exactly one wine that he/she likes. Let the class *Guest* indicate the persons that will be invited to the dinner and let the class *SelectedWine* indicate the wines chosen to be served. An ERDF program P that describes this wine selection problem is the following (commas "," in the body of the rules indicate conjunction $\wedge$):

$$id(?x, ?x) \ \leftarrow\ rdf{:}type(?x, rdfs{:}Resource).$$
$$rdf{:}type(?y, SelectedWine) \ \leftarrow rdf{:}type(?x, Guest), rdf{:}type(?y, Wine), likes(?x, ?y),$$
$$\forall ?z\ (rdf{:}type(?z, SelectedWine), {\sim}id(?z, ?y)\ \supset\ {\sim}likes(?x, ?z)).$$

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

$$G = \{\ rdf{:}type(Carlos, Guest),\ rdf{:}type(Gerd, Guest),\ rdf{:}type(Riesling, Wine),$$
$$rdf{:}type(Retsina, Wine),\ rdf{:}type(Chardonnay, Wine),\ likes(Gerd, Riesling),$$
$$likes(Gerd, Retsina),\ likes(Carlos, Chardonnay),\ likes(Carlos, Retsina)\ \}.$$

Then, according to Definition 5.1, the ERDF ontology $O = \langle G, P\rangle$ has two stable models, M_1 and M_2, such that:

$$M_1 \models rdf{:}type(Riesling, SelectedWine) \ \wedge \ rdf{:}type(Chardonnay, SelectedWine) \ \wedge \\ \sim rdf{:}type(Retsina, SelectedWine).$$

$$M_2 \models rdf{:}type(Retsina, SelectedWine) \ \wedge \ \sim rdf{:}type(Riesling, SelectedWine) \ \wedge \\ \sim rdf{:}type(Chardonnay, SelectedWine).$$

Note that, according to stable model M_1, the wines selected for the dinner are *Riesling* and *Chardonnay*. This is because, (i) *Gerd* likes *Riesling* but does not like *Chardonnay*, and (ii) *Carlos* likes *Chardonnay* but does not like *Riesling*.

According to stable model M_2, only *Retsina* is selected for the dinner. This is because, both *Gerd* and *Carlos* like *Retsina*.

Stable model M_1 is reached through the chain $I_0 \leq M_1 \leq M_1$, where I_0 is the single Herbrand interpretation in $minimal(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})$. To verify this, note that:

$$P_{[I_0,M_1]} = P_{[M_1,M_1]} =$$

$$[id(?x,?x) \ \leftarrow \ rdf{:}type(?x, rdfs{:}Resource)]_{V_O} \cup$$

$$\{rdf{:}type(Riesling, SelectedWine) \ \leftarrow \ rdf{:}type(Gerd, Guest), \\ rdf{:}type(Riesling, Wine), likes(Gerd, Riesling), \\ \forall ?z \ (rdf{:}type(?z, SelectedWine), {\sim}id(?z, Riesling) \ \supset \ {\sim}likes(Gerd, ?z))\} \cup$$

$$\{rdf{:}type(Chardonnay, SelectedWine) \ \leftarrow rdf{:}type(Carlos, Guest), \\ rdf{:}type(Chardonnay, Wine), likes(Carlos, Chardonnay), \\ \forall ?z \ (rdf{:}type(?z, SelectedWine), {\sim}id(?z, Chardonnay) \ \supset \ {\sim}likes(Carlos, ?z))\}.$$

Similarly, stable model M_2 is reached through the chain $I_0 \leq M_2 \leq M_2$. To verify this, note that:

$$P_{[I_0,M_2]} = P_{[M_2,M_2]} =$$

$$[id(?x,?x) \ \leftarrow \ rdf{:}type(?x, rdfs{:}Resource)]_{V_O} \cup$$

$$\{rdf{:}type(Retsina, SelectedWine) \leftarrow \ rdf{:}type(Gerd, Guest), \\ rdf{:}type(Retsina, Wine), likes(Gerd, Retsina), \\ \forall ?z \ (rdf{:}type(?z, SelectedWine), {\sim}id(?z, Retsina) \ \supset \ {\sim}likes(Gerd, ?z))\} \cup$$

$$\{rdf{:}type(Retsina, SelectedWine) \leftarrow rdf{:}type(Carlos, Guest), \\ rdf{:}type(Retsina, Wine), likes(Carlos, Retsina), \\ \forall ?z \ (rdf{:}type(?z, SelectedWine), {\sim}id(?z, Retsina) \ \supset \ {\sim}likes(Carlos, ?z))\}.$$

Assume now that *Retsina* should not be one of the selected wines, because it does not match with the food. To indicate this, we add to P the ERDF constraint:

$$false \leftarrow rdf{:}type(Retsina, SelectedWine).$$

Then, M_1 is the single model of the modified ontology.

It is easy to verify that if O is an ERDF ontology and O' is exactly as O, but without the ERDF constraints appearing in O, then $\mathcal{M}^{st}(O) \subseteq \mathcal{M}^{st}(O')$. In other words, the ERDF constraints appearing in an ERDF ontology eliminate undesirable stable models.

Paper Assignment Example: Consider a class *Paper* whose instances are papers submitted to a conference, a class *Reviewer* whose instances are potential reviewers for the submitted papers, and a property *conflict*(R, P) indicating that there is a conflict of interest between reviewer R and paper P. Assume now that we want to assign papers to reviewers based 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) a paper is not assigned to a reviewer, if there is a conflict of interest. The assignment of a paper P to a reviewer R is indicated through the property *assign*(P, R). The ERDF triple *allAssigned*(*Paper*, *Reviewer*) indicates that each paper has been assigned to one reviewer. An ERDF program P describing the assignment of papers is the following:

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

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

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

Then, according to Definition 5.1, the ERDF ontology $O = \langle G, P \rangle$ has four stable models, denoted by M_1, ..., M_4, such that:

$$
\begin{aligned}
M_1 &\models assign(P1, R1) \wedge assign(P2, R3) \wedge {\sim}allAssigned(Paper, Reviewer), \\
M_2 &\models assign(P1, R1) \wedge assign(P3, R3) \wedge {\sim}allAssigned(Paper, Reviewer), \\
M_3 &\models assign(P1, R2) \wedge assign(P2, R1) \wedge assign(P3, R3) \wedge allAssigned(Paper, Reviewer), \\
M_4 &\models assign(P1, R2) \wedge assign(P2, R3) \wedge assign(P3, R1) \wedge allAssigned(Paper, Reviewer).
\end{aligned}
$$

We would like to note that, in contrast to the previous examples, given an ERDF ontology $O = \langle G, P \rangle$, it is possible that $|minimal(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})| > 1$, due to the declaration of total properties and total classes. Specifically, the number of the interpretations I_0 in item 1 of Definition 5.1 is more than one iff G contains ERDF triples of the form *rdf*:*type*(*p*, *erdf*:*TotalProperty*) or *rdf*:*type*(*c*, *erdf*:*TotalClass*). For example, let $O = \langle G, \emptyset \rangle$, where:

$$G = \{authorOf(John, book1), authorOf(Peter, book2),\ rdf{:}type(authorOf, erdf{:}TotalProperty)\}.$$

Then, there are $I_0, I'_0 \in minimal(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})$ such that:

$$I_0 \models authorOf(John, book2) \quad \text{and} \quad I'_0 \models \neg authorOf(John, book2).$$

Note that both I_0 and I'_0 are stable models of O. However, I_0 satisfies *authorOf*(*John*, *book2*), even though there is no evidence that *John* is an author of *book2*.

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

Proposition 5.1 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[15]. Obviously, this is a desirable result since, in this case, an open-world assumption is made for all properties. Thus, there is no preferential entailment of weak negation, for any of the properties. Of course, the term "stable model" is not very descriptive, for this degenerative case.

Proposition 5.2 Let $O = \langle G, P \rangle$ be an ERDF ontology such that $\mathit{rdfs{:}subClassOf}(\mathit{rdf{:}Property}, \mathit{erdf{:}TotalProperty}) \in G$. Then, $\mathcal{M}^{st}(O) = \mathcal{M}^H(O)$.

A final note is that, similarly to stable models defined by Gelfond & Lifschitz (1988, 1990) and Herre et al. (1999), ERDF 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 $p, s, o \in \mathcal{URI}$. Then, $\mathcal{M}^{st}(O) = \emptyset$, whereas there is a Herbrand model of O that satisfies $p(s, o)$.

6. ERDF Stable Model Entailment & Stable Answers

In this section, we define stable model entailment on ERDF ontologies, showing that it extends ERDF entailment on ERDF graphs. Moreover, we define the skeptical and credulous answers of an ERDF formula (query) F w.r.t. an ERDF ontology O.

Definition 6.1 (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 $M \in \mathcal{M}^{st}(O)$, $\quad M \models F$. $\square$

For example, let $O = \langle \emptyset, P \rangle$, where $P = \{p(s, o) \leftarrow {\sim}q(s, o)\}$ and $p, q, s, o \in \mathcal{URI}$. Then, $O \models^{st} {\sim}q(s, o) \wedge p(s, o)$.

Now, let $G = \{\mathit{rdfs{:}subClassOf}(\mathit{rdf{:}Property}, \mathit{erdf{:}TotalProperty})\}$ and let P be as in the previous example. Then, $\langle G, P \rangle \models^{st} q(s, o) \vee p(s, o)$, but $\langle G, P \rangle \not\models^{st} {\sim}q(s, o)$ and $\langle G, P \rangle \not\models^{st} p(s, o)$. Note that this is the desirable result, since now q is a total property (and thus, an open-world assumption is made for q).

As another example, let $p, s, o \in \mathcal{URI}$, 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).

Now, let $G = \{\mathit{rdf{:}type}(p, \mathit{erdf{:}TotalProperty}), p(s, o)\}$ 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 3.1), 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.

EU Membership Example: Consider the following ERDF program P, specifying some rules for concluding that a country is not a member state of the European Union (EU).

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

15. Note that, in this case, $M \in \mathit{minimal}(\{I \in \mathcal{I}^H(O) \mid I \models \mathit{sk}(G)\})$ and $M \in \mathit{minimal}(\{I \in \mathcal{I}^H(O) \mid I \geq M$ and $I \models \mathit{Concl}(r),\ \forall\, r \in P_{[M,M]}\})$.

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

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

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 $rdf{:}type(Italy, EUMember)$). Thus, incorrect information is obtained by the closed-world assumption expressed in rule r_2. In the case that $rdf{:}type(EUMember, erdf{:}TotalClass)$ is added to G (that is, an open-world assumption is made for the class $EUMember$) then $\sim rdf{:}type(Italy, EUMember)$ and thus, $\neg rdf{:}type(Italy, EUMember)$ are no longer entailed. This is because, there is a stable model of the extended ERDF ontology O that satisfies $rdf{:}type(Italy, EUMember)$. 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 the ERDF triple $rdf{:}type(Italy, EUMember)$.

The following proposition follows directly from the fact that any stable model of an ERDF ontology O is an ERDF interpretation.

Proposition 6.1 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 $O \models^{st} F'$.

For ERDF graphs G, G', it can be proved that $\langle G, \emptyset \rangle \models^{st} G'$ iff $G \models^{ERDF} G'$ (see below). Now the question arises whether this result can be generalized by replacing the ERDF graph G' by any ERDF formula F. The following example shows that this is not the case. Let $G = \{p(s, o)\}$ and let $F = \sim p(o, s)$, where $p, s, o \in \mathcal{URI}$. Then $\langle G, \emptyset \rangle \models^{st} F$, whereas $G \not\models^{ERDF} F$. However, G' can be replaced by any ERDF d-formula F, defined as follows:

Definition 6.2 (ERDF d-formula) Let F be an ERDF formula. We say that F is an *ERDF d-formula* iff (i) F is the disjunction of existentially quantified conjunctions of ERDF triples, and (ii) $FVar(F) = \emptyset$. $\Box$

For example, let:

$$F = (\exists ?x \; rdf{:}type(?x, \mathit{Vertex}) \wedge rdf{:}type(?x, \mathit{Red})) \vee \\ (\exists ?x \; rdf{:}type(?x, \mathit{Vertex}) \wedge rdf{:}type(?x, \mathit{Blue})).$$

Then, F is an ERDF d-formula. It is easy to see that if G is an ERDF graph then $formula(G)$ is an ERDF d-formula.

Proposition 6.2 Let G be an ERDF graph and let F be an ERDF formula such that $V_F \cap sk_G(Var(G)) = \emptyset$. It holds:

1. If F is an ERDF d-formula and $\langle G, \emptyset \rangle \models^{st} F$ then $G \models^{ERDF} F$.
2. If $G \models^{ERDF} F$ then $\langle G, \emptyset \rangle \models^{st} F$.

Let G be an ERDF graph and let F be an ERDF d-formula or an ERDF graph such that $V_F \cap sk_G(Var(G)) = \emptyset$. A direct consequence of Proposition 6.2 is that:

$$\langle G, \emptyset \rangle \models^{st} F \text{ iff } G \models^{ERDF} F.$$

The following proposition is a direct consequence of Proposition 3.2 and Proposition 6.2, and shows that stable model entailment extends RDFS entailment from RDF graphs to ERDF ontologies.

Proposition 6.3 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 $\langle G, \emptyset \rangle \models^{st} G'$.

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 6.3 is actually trivial.

Definition 6.3 (ERDF query, ERDF stable answers) Let $O = \langle G, P \rangle$ be an ERDF ontology. An *(ERDF) query* F is an ERDF formula. The *(ERDF) stable answers* of F w.r.t. O are defined as follows:

$$Ans_O^{st}(F) = \begin{cases} \text{"yes"} & \text{if } FVar(F) = \emptyset \text{ and } \forall M \in \mathcal{M}^{st}(O) : M \models F \\ \text{"no"} & \text{if } FVar(F) = \emptyset \text{ and } \exists M \in \mathcal{M}^{st}(O) : M \not\models F \\ \{v : FVar(F) \rightarrow V_O \mid \forall M \in \mathcal{M}^{st}(O),\ M \models v(F)\} & \text{if } FVar(F) \neq \emptyset, \end{cases}$$

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

For example, let $p, q, c, s, o \in \mathcal{URI}$, 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 $Ans_O^{st}(F) = \{\{?x = o, ?y = o\}, \{?x = s, ?y = s\}, \{?x = o, ?y = s\}\}$.

Let $O = \langle G, P \rangle$, where $q, s, o \in \mathcal{URI}$, $G = \{rdf{:}type(p, erdf{:}TotalProperty), q(s, o)\}$, and $P = \{\neg p(?x, ?y) \leftarrow {\sim}p(?x, ?y)\}$. Then, it holds $Ans_O^{st}(p(?x, ?y)) = Ans_O^{st}({\sim}p(?x, ?y)) = Ans_O^{st}(\neg p(?x, ?y)) = \emptyset$. This is because, in contrast to the above example, p is a total property. Thus, for all mappings $v : \{?x, ?y\} \rightarrow V_O$, 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 $M' \models v({\sim}p(?x, ?y) \wedge \neg p(?x, ?y))$.

Consider the ERDF ontology O of the paper assignment example, below Definition 5.1. Then, $Ans_O^{st}(assign(P1, R2)) =$"yes" and $Ans_O^{st}(assign(P2, R1)) =$"no". Though $Ans_O^{st}(assign(P2, R1)) =$"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 $allAssigned(Paper, Reviewer)$, indicating that the desirable paper assignment has been achieved.

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

Definition 6.4 (Credulous ERDF 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:

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

where $ans_M(F) = \{v : FVar(F) \to V_O \mid M \models v(F)\}$. $\square$

Continuing with the paper assignment example, consider the query:

$$F = allAssigned(Paper, Reviewer).$$

Then, although $Ans^{st}_O(F) =$“no”, it holds $c\text{-}Ans^{st}_O(F) =$“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,

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

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^H_O, 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:

$$G = \{rdf{:}type(x, c1) \mid x \in \{c1, c2\} \cup V'\},$$

where $V' = (\mathcal{V}_{RDF} - \{rdf{:}\_i \mid i \in I\!N\}) \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Additionally, consider the ERDF program:

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

Let $F = \forall ?x\ rdf{:}type(?x, c2) \supset rdf{:}type(?x, c1)$. 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 $M, v \models rdf{:}type(?x, c2)$ and $M, v \not\models rdf{:}type(?x, c1)$.

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

$$G = \{teaches(Anne, CS301), teaches(Peter, CS505), rdf{:}type(CS505, GradCourse)\}.$$

Let $F = \forall ?x \; teaches(Peter, ?x) \supset rdf{:}type(?x, GradCourse)$. 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 $v{:}\{?x\} \rightarrow D$ such that $M, v \models teaches(Peter, ?x)$ and $M, v \not\models rdf{:}type(?x, GradCourse)$. In other words, FOL makes an open-world assumption for *teaches*.

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

This example also shows that, in contrast to FOL, stable model entailment is non-monotonic.

Note that the previous ERDF graph G can also be seen as a Description Logic A-Box A (Baader et al., 2003), where

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

Consider a T-Box $T = \emptyset$. Since Description Logics (DLs) are 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 by Donini et al. (2002), 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 graduate courses. Note that this conclusion is non-monotonic, and 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. An XML-based Syntax for ERDF

A natural approach to define an XML syntax for ERDF is: (i) to follow the RDF/XML syntax (Beckett, 2004), as much as possible, and (ii) to extend it in a suitable way, where necessary. Following this approach, we briefly present here an XML syntax for ERDF. Details are going to be given in a subsequent paper.

Classes and properties are defined with the help of the `rdfs:Class` and `rdf:Property` elements of the RDF/XML syntax. Similarly, total classes and total properties are defined with the help of the `erdf:TotalClass` and `erdf:TotalProperty` elements of the ERDF/XML syntax.

Example 7.1 The following ERDF/XML statements:

```
<rdf:Property rdf:about="#likes">
  <rdfs:domain rdf:resource="#Person"/>
```

```
</rdf:Property>

<erdf:TotalProperty rdf:about="#authorOf">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Book"/>
</erdf:TotalProperty>
```

correspond to the ERDF graph:

$$G = \{\ rdf{:}type(likes, rdf{:}Property), rdfs{:}domain(likes, Person),\\ rdf{:}type(authorOf, erdf{:}TotalProperty), rdfs{:}domain(authorOf, Person),\\ rdfs{:}range(authorOf, Book)\}.$$

ERDF triples (and sets of ERDF triples sharing the same subject term) are encoded by means of the `erdf:Description` element. Each description contains a non-empty list of (possibly negated) property-value slots *about* the subject term.

- URI references, blank node identifiers, and variables that appear in the *subject* position of an ERDF triple are expressed as values of the `erdf:about` attribute, using the SPARQL syntax (Prud'hommeaux & Seaborne, 2008) for blank node identifiers and variables. On the other hand, literals that appear in the subject position of an ERDF triple are expressed as the text content of the `erdf:about` subelement.
- URI references, blank node identifiers, and variables that appear in the *object* position of an ERDF triple are expressed as values of the attributes `rdf:resource`, `rdf:nodeID`, and `erdf:variable`, respectively. On the other hand, literals that appear in the object position of an ERDF triple are expressed as the text content of the corresponding property subelement.

Example 7.2 The following `erdf:Description` statements:

```
<erdf:Description erdf:about="#Gerd">
  <ex:authorOf rdf:nodeID="x"/>
  <ex:likes rdf:resource="#Chicken"/>
  <ex:likes erdf:negationMode="Sneg" rdf:resource="#Pork"/>
</erdf:Description>

<erdf:Description>
  <erdf:About rdf:datatype="&xsd;string">Grigoris</erdf:About>
  <ex:denotationOf rdf:resource="#Grigoris"/>
</erdf:Description>
```

correspond to the ERDF graph:

$$G = \{\ authorOf(Gerd, ?x), likes(Gerd, Chicken), \neg likes(Gerd, Pork),\\ denotationOf(\text{“}Grigoris\text{”}\hat{}\hat{}xsd{:}string, Grigoris)\ \}.$$

Now, in order to express ERDF rules with XML, we use the rule markup language R2ML (*REWERSE Rule Markup Language*) (Wagner, Giurca, & Lukichev, 2006, 2005), which is a general XML-based markup language for representing derivation rules and integrity constraints. This is demonstrated in the following example:

Example 7.3 The following `erdf:DerivationRule` statement:

```
<r2ml:DerivationRule r2ml:ruleID="R1">
  <r2ml:conditions>
    <erdf:Description erdf:about="?x">
      <rdf:type rdf:resource="#MainDish"/>
    </erdf:Description>
    <erdf:Description erdf:about="?y">
      <rdf:type rdf:resource="#Guest"/>
      <ex:likes erdf:variable="x"/>
    </erdf:Description>
    <r2ml:NegationAsFailure>
      <r2ml:ExistentiallyQuantifiedFormula>
        <r2ml:GenericVariable r2ml:name="z" r2ml:class="#Guest"/>
        <erdf:Description erdf:about="?z">
          <ex:likes erdf:negationMode="Sneg" erdf:variable="x"/>
        </erdf:Description>
      </r2ml:ExistentiallyQuantifiedFormula>
    </r2ml:NegationAsFailure>
  </r2ml:conditions>
  <r2ml:conclusion>
    <erdf:Description erdf:about="?x">
      <rdf:type rdf:resource="#SelectedMainDish"/>
    </erdf:Description>
  </r2ml:conclusion>
</r2ml:DerivationRule>
```

expresses that a main dish is selected for dinner, if there is a guest who likes it and no guest who dislikes it. Specifically, it corresponds to the ERDF rule:

$$\begin{aligned} rdf{:}type(?x, SelectedMainDish) \leftarrow\; & rdf{:}type(?x, MainDish), rdf{:}type(?y, Guest), likes(?y, ?x), \\ & \sim (\exists ?z\ rdf{:}type(?z, Guest), \neg likes(?z, ?x)). \end{aligned}$$

8. Undecidability of the ERDF Stable Model Semantics

The main difficulty in the computation of the ERDF stable model semantics is the fact that $\mathcal{V}_{RDF}$ is infinite, and thus the vocabulary of any ERDF ontology O is also infinite (note that $\{rdf{:}\_i \mid i \in I\!N\} \subseteq \mathcal{V}_{RDF} \subseteq V_O$). Due to this fact, satisfiability and entailment under the ERDF stable model semantics are in general undecidable.

The proof of undecidability exploits a reduction from the *unbounded tiling problem*. The unbounded tiling problem consists in placing tiles on an infinite grid, satisfying a given set of constraints on adjacent tiles. Specifically, the unbounded tiling problem is a structure $\mathcal{D} = \langle \mathcal{T}, H, V\rangle$, where $\mathcal{T} = \{T_1, ..., T_n\}$ is a finite set of tile types and $H, V \subseteq \mathcal{T} \times \mathcal{T}$ specify which tiles can be adjacent horizontally and vertically, respectively. A *solution* to $\mathcal{D}$ is a *tiling*, that is, a total function $\tau : I\!N \times I\!N \to \mathcal{T}$ such that: $(\tau(i, j), \tau(i + 1, j)) \in H$ and $(\tau(i, j), \tau(i, j + 1)) \in V$, for all $i, j \in I\!N$. The existence of a solution for a given unbounded tiling problem is known to be undecidable (Berger, 1966).

Let $\mathcal{D} = \langle \mathcal{T}, H, V\rangle$ be an instance of the unbounded tiling problem, where $\mathcal{T} = \{T_1, ..., T_n\}$. We will construct an ERDF ontology $O_{\mathcal{D}} = \langle G, P\rangle$ and an ERDF formula $F_{\mathcal{D}}$ such that $\mathcal{D}$ has a solution iff $O_{\mathcal{D}}$ does not entail $F_{\mathcal{D}}$ under the ERDF stable model semantics.

Consider (i) a class *Tile* whose instances are the tiles placed on the infinite grid, (ii) a property $right(x, y)$ indicating that tile y is right next to tile x, (iii) a property $above(x, y)$ indicating that tile y is exactly above tile x, (iv) a class *HasRight* whose instances are the tiles for which there exists a tile right next to them, (v) a class *HasAbove* whose instances are the tiles for which there exists a tile exactly above them, (vi) a property $Type(x, T)$, indicating that the type of tile x is T, (vii) a property $HConstraint(T, T')$, indicating that $(T, T') \in H$, and (viii) a property $VConstraint(T, T')$, indicating that $(T, T') \in V$.

Let G be the ERDF graph:

$$
\begin{aligned}
G = \ & \{rdfs{:}subClassOf(rdfs{:}ContainerMembershipProperty, Tile),\\
& \ rdfs{:}subClassOf(Tile, rdfs{:}ContainerMembershipProperty)\} \cup\\
& \{HConstraint(T, T') \mid (T, T') \in H\} \ \cup \ \{VConstraint(T, T') \mid (T, T') \in V\} \ .
\end{aligned}
$$

Let P be the ERDF program, containing the following rules (and constraints):

(1) $Type(?x, T_1) \leftarrow rdf{:}type(?x, Tile), \sim Type(?x, T_2), ..., \sim Type(?x, T_n).$

$$
\begin{aligned}
Type(?x, T_i) \leftarrow \ & rdf{:}type(?x, Tile), \sim Type(?x, T_1), ..., \sim Type(?x, T_{i-1}),\\
& \sim Type(?x, T_{i+1}), ..., \sim Type(?x, T_n), \text{ for all } i = 2, ..., n-1.
\end{aligned}
$$

$Type(?x, T_n) \leftarrow rdf{:}type(?x, Tile), \sim Type(?x, T_1), ..., \sim Type(?x, T_{n-1}).$

(2)
$$
\begin{aligned}
right(?x, ?y) & \leftarrow rdf{:}type(?x, Tile), rdf{:}type(?y, Tile), \sim\neg right(?x, ?y).\\
\neg right(?x, ?y) & \leftarrow rdf{:}type(?x, Tile), rdf{:}type(?y, Tile), \sim right(?x, ?y).
\end{aligned}
$$

(3)
$$
\begin{aligned}
above(?x, ?y) & \leftarrow rdf{:}type(?x, Tile), rdf{:}type(?y, Tile), \sim\neg above(?x, ?y).\\
\neg above(?x, ?y) & \leftarrow rdf{:}type(?x, Tile), rdf{:}type(?y, Tile), \sim above(?x, ?y).
\end{aligned}
$$

(4)
$$
\begin{aligned}
rdf{:}type(?x, HasRight) & \leftarrow right(?x, ?y).\\
rdf{:}type(?x, HasAbove) & \leftarrow above(?x, ?y).
\end{aligned}
$$

$$
\begin{aligned}
false & \leftarrow rdf{:}type(?x, Tile), \sim rdf{:}type(?x, HasRight).\\
false & \leftarrow rdf{:}type(?x, Tile), \sim rdf{:}type(?x, HasAbove).
\end{aligned}
$$

$id(?x, ?x) \leftarrow rdf{:}type(?x, rdfs{:}Resource).$

$$
\begin{aligned}
false & \leftarrow right(?x, ?y), right(?x, ?y'), \sim id(?y, ?y').\\
false & \leftarrow above(?x, ?y), above(?x, ?y'), \sim id(?y, ?y').
\end{aligned}
$$

(5)
$$
\begin{aligned}
false & \leftarrow right(?x, ?y), Type(?x, ?T), Type(?y, ?T'), \sim HConstraint(?T, ?T').\\
false & \leftarrow above(?x, ?y), Type(?x, ?T), Type(?y, ?T'), \sim VConstraint(?T, ?T').
\end{aligned}
$$

Note that in all stable models of $O_{\mathcal{D}} = \langle G, P \rangle$, the class *Tile* contains exactly the (infinite in mumber) $rdf{:}\_i$ terms, for $i \in I\!N$. This is because, computing the stable models of O, only the minimal models of $sk(G)$ are considered (see Definition 5.1, Step 1). Thus, each tile on the infinite grid is represented by an $rdf{:}\_i$ term, for $i \in I\!N$.

Intuitively, rule set (1) expresses that each tile should have exactly one associated type in $\mathcal{T}$. Rule set (2) expresses that two tiles are either horizontally adjacent on the grid or not horizontally adjacent. Rule set (3) expresses that two tiles are either vertically adjacent on the grid or not vertically adjacent. Rule set (4) expresses that each tile should have exactly one tile right next to it and exactly one tile right above it. Rule set (5) expresses that the types of horizontally and vertically adjacent tiles should respect the H and V relations of $\mathcal{D}$, respectively.

To finalize the reduction, we define:

$$F_{\mathcal{D}} = \exists ?x, \exists ?y, \exists ?x', \exists ?y', \exists ?x'' \quad right(?x, ?y) \wedge above(?y, ?y') \wedge right(?x', ?y') \wedge above(?x'', ?x') \wedge \sim id(?x, ?x'').$$

Formula $F_{\mathcal{D}}$ expresses that there is a tile x such that, starting from x, if we move:

one step right $\rightarrow$ *one step up* $\rightarrow$ *one step left* $\rightarrow$ *one step down*

then we will meet a tile x'' different than x.

Proposition 8.1 Let $\mathcal{D}$ be an instance of the unbounded tiling problem. It holds:

1. $\mathcal{D}$ has a solution iff $O_{\mathcal{D}} \cup \{false \ \leftarrow \ F_{\mathcal{D}}\}$ has a stable model.
2. $\mathcal{D}$ has a solution iff $O_{\mathcal{D}} \not\models^{st} F_{\mathcal{D}}$.

Since the unbounded tiling problem is undecidable (Berger, 1966), it follows directly from Proposition 8.1 that satisfiability and entailment under the ERDF stable model semantics are in general undecidable.

The previous reduction shows that both problems remain undecidable for an ERDF ontology $O = \langle G, P \rangle$, even if (i) the body of each rule in P has the form $t_1, ..., t_k, \sim t_{k+1}, ..., \sim t_n$, where t_i is an ERDF triple and (ii) the terms *erdf:TotalClass* and *erdf:TotalProperty* do not appear in O, that is, $(V_G \cup V_P) \cap \mathcal{V}_{ERDF} = \emptyset$. Note that since each constraint $false \leftarrow F$ that appears in an ERDF ontology O can be replaced by the rule $\neg t \leftarrow F$, where t is an RDF, RDFS, or ERDF axiomatic triple, the presence of constraints in O does not affect decidability.

Future work concerns the identification of syntactic restrictions for an ERDF ontology O such that ERDF stable model entailment is decidable.

9. 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 non-monotonic inference, simply by not considering all models of a rule as being intended, but only those models that satisfy certain criteria. Thus, the stable model semantics for normal and (generalized) extended logic programs (Gelfond & Lifschitz, 1988, 1990; Herre & Wagner, 1997; Herre et al., 1999) can be viewed as a Tarski-style model-theoretic semantics for non-monotonic derivation rules.

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

- $\mathcal{L}$ is a set of formulas, called *language*,

- $\mathcal{I}$ is a set of interpretations, and
- $\models$ is a relation between interpretations and formulas, called *model relation.*

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

- a notion of *derivation rule* $G \leftarrow F$, where $F \in \mathcal{L}$ is called *condition* and $G \in \mathcal{L}$ is called *conclusion*,
- a set of derivation rules $\mathcal{DR}_{\mathcal{L}} = \{G \leftarrow F \mid F, G \in \mathcal{L}\}$,
- an extension of the model relation $\models$ to include also pairs of interpretations and derivation rules, and
- a standard model operator $\mathcal{M}(KB) = \{I \in \mathcal{I} \mid I \models X,\ \forall X \in KB\}$, where $KB \subseteq \mathcal{L} \cup \mathcal{DR}_{\mathcal{L}}$ 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 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_1 *is less informative* than I_2. An *information model theory* $\langle \mathcal{L},\ \mathcal{I},\ \models,\ \leq \rangle$ is a Tarski-style model theory, extended by an information ordering $\leq$.

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 (Gelfond & Lifschitz, 1988, 1990; Herre & Wagner, 1997; Herre et al., 1999).

For any given model operator $\mathcal{M}^x : \mathcal{P}(\mathcal{L} \cup \mathcal{DR}_{\mathcal{L}}) \rightarrow \mathcal{P}(\mathcal{I})$, knowledge base $KB \subseteq \mathcal{L} \cup \mathcal{DR}_{\mathcal{L}}$, and $F \in \mathcal{L}$, 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 *non-monotonic*, 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 $\mathcal{L} = L(\mathcal{URI} \cup \mathcal{LIT})$, $\mathcal{I}$ is the set of ERDF interpretations over any vocabulary $V \subseteq \mathcal{URI} \cup \mathcal{LIT}$, and the model relation $\models$ is as defined in Definitions 3.5 and 4.3. In our theory, the intended model operator ($\mathcal{M}^{st}$) assigns to each ERDF ontology a (possibly empty) set of stable models (Definition 5.1).

10. Related Work

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

Ter Horst (2005b, 2004) generalizes RDF graphs to *generalized RDF graphs*, by allowing variables in the property position of RDF triples. Additionally, the author extends the RDFS semantics with datatypes and part of the OWL vocabulary, defining the $pD*$ semantics, which extends the "if-semantics" of RDFS and is weaker than the "iff-semantics" of D-entailment (Hayes, 2004) and OWL Full (Patel-Schneider, Hayes, & Horrocks, 2004). A sound and complete set of entailment rules for $pD*$ entailment is also presented.

In a subsequent work, ter Horst (2005a) considers the extension of the previous framework with the inclusion of rules of the form "if G then G'", where G is an RDF graph without blank nodes but possibly with variables and G' is a generalized RDF graph, possibly with both blank nodes and variables. Intuitively, rule variables are universally quantified in the front of the rule (like the free variables of our rules) and blank nodes in the head of the rule correspond to existentially quantified variables (this feature is not supported in our model). Based on a set of rules R and a datatype map D, R-entailment[16] is defined between two generalized RDF graphs G and G' ($G \models_R G'$), and a set of sound and complete rules for R-entailment is presented. To relate our work with that of ter Horst (2005a), we state the following proposition:

Let D be a datatype map, containing only *rdf*:*XMLLiteral*, and let R be a set of rules of the form "if G then G'" with the constraints: (i) all terms appearing in property position are URIs, (ii) if $G \neq \{\}$ then no blank node appears in G', and (iii) $V_R \cap (\mathcal{V}_{pOWL} \cup \mathcal{V}_{ERDF}) = \emptyset$, where $\mathcal{V}_{pOWL}$ denotes the part of the OWL vocabulary, included in the $pD*$ semantics. Let G, G' be RDF graphs such that $(V_G \cup V_{G'}) \cap (\mathcal{V}_{pOWL} \cup \mathcal{V}_{ERDF}) = \emptyset$. Then based on G and R, we can define, by a simple transformation, an ERDF ontology O such that $G \models_R G'$ iff $O \models^{st} G'$.

However, in this work, weak and strong negation are not considered. Thus, closed-world reasoning is not supported. Additionally, in our theory, the condition of a rule is any ERDF formula over a vocabulary V, (thus, involving any of the logical factors $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$, $\forall$, and $\exists$), and not just a conjunction of positive triples.

TRIPLE (Sintek & Decker, 2002) 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 (Kifer, Lausen, & Wu, 1995) 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 s, 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 the rules is restricted to 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. TRIPLE uses the Lloyd-Topor transformations (Lloyd & Topor, 1984) to take care of the first-order connectives in the

16. The symbol D does not appear explicitly in the notation of R-entailement, for reasons of simplification.

sentences and supports weak negation under the well-founded semantics (Gelder, Ross, & Schlipf, 1991). Strong negation is not used.

Flora-2 (Yang, Kifer, & Zhao, 2003) is a rule-based object-oriented knowledge base system for reasoning with semantic information on the Web. It is based on F-logic (Kifer et al., 1995) and supports metaprogramming, non-monotonic multiple inheritance, logical database updates, encapsulation, dynamic modules, and two kinds of weak negation. Specifically, it supports Prolog negation and well-founded negation (Gelder et al., 1991), through invocation of the corresponding operators \+ and *tnot* of the XSB system (Rao, Sagonas, Swift, Warren, & Freire, 1997). The formal semantics for non-monotonic multiple inheritance is defined by Yang & Kifer (2003a). In addition, Flora-2 supports reification and anonymous resources (Yang & Kifer, 2003b). 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) (Berners-Lee, Connolly, Kagal, Scharf, & Hendler, 2008) provides a more human readable syntax for RDF and also extends RDF by adding numerous predefined constructs ("built-ins") for being able to express rules conveniently. Remarkably, N3 contains a built-in (`log:definitiveDocument`) for making restricted completeness assumptions and another built-in (`log:notIncludes`) for expressing simple negation-as-failure tests. The addition of these constructs was motivated by use cases. However, N3 does not provide strong negation and closed-world reasoning is not fully supported. N3 is supported by the CWM system[17], a forward engine especially designed for the Semantic Web, and the Euler system[18], a backward engine relying on loop checking techniques to guarantee termination.

Alferes et al. (2003) propose the paraconsistent well-founded semantics with explicit negation ($WFSX_P$)[19], 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. No formal model theory has been explicitly provided for the integrated logic.

DR-Prolog (Antoniou, Bikakis, & Wagner, 2004) and DR-DEVICE (Bassiliades, Antoniou, & Vlahavas, 2004) 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 (Antoniou, Billington, Governatori, & Maher, 2001; Maher, 2002). Defeasible logic contains only one kind of negation (strong negation) in the object language[20] and allows to reason in the presence of contradiction and incomplete information. It supports

17. http://www.w3.org/2000/10/swap/doc/cwm.html.
18. http://www.agfa.com/w3c/euler/.
19. $WFSX_P$ (Alferes, Damásio, & Pereira, 1995) is an extension of the well-founded semantics with explicit negation (WFSX) on extended logic programs (Pereira & Alferes, 1992) and, thus, also of the well-founded semantics (WFS) on normal logic programs (Gelder et al., 1991).
20. However, in defeasible logic, negation-as-failure can be easily simulated by other language ingredients.

monotonic and non-monotonic rules, exceptions, default inheritance, and preferences. No formal model theory has been explicitly provided for the integrated logic.

OWL-DL (McGuinness & van Harmelen, 2004) 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 (Horrocks & Patel-Schneider, 2003). 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 (Horrocks, Patel-Schneider, Boley, Tabet, Grosof, & Dean, 2004). Horrocks & Patel-Schneider (2004) show that this extension is in general undecidable. $\mathcal{AL}$-log (Donini, Lenzerini, Nardi, & Schaerf, 1998) 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 (Levy & Rousset, 1998) 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)[21]. Motik et al. (2004) show 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 algorithm is employed for query answering but L is translated to a disjunctive logic program $DD(L)$ which is combined with P for answering ground queries.

In this category of works, entailment on the DL, that has been extended with rules, is based on first-order logic. This means that 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, our work supports both weak and strong negation, and allows closed-world and open-world reasoning on a selective basis.

A different kind of integration is achieved by Eiter et al. (2004a). In this work, 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 to L and vice-versa. The answer set semantics of $\langle L, P \rangle$ are defined, as a generalization of the answer set semantics (Gelfond & Lifschitz, 1990) on ordinary extended logic programs. A similar kind of integration is achieved by Eiter et al. (2004b). In this work, 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, as a

21. 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.

generalization of the well-founded semantics (Gelder et al., 1991) of ordinary normal logic programs. Obviously, in both of these works, 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.

Rosati (1999) defines the semantics of a *disjunctive* $\mathcal{AL}$-log knowledge base, based on the stable model semantics for disjunctive databases (Gelfond & Lifschitz, 1991), extending $\mathcal{AL}$-log (Donini et al., 1998). A disjunctive $\mathcal{AL}$-log knowledge base is the integration of an $\mathcal{ALC}$ knowledge base T with a (safe) disjunctive logic program P that allows concept and role DL-atoms in the body of the rules (along with weak negation on non DL-atoms). The safety condition enforces that each variable in the head of a rule should also appear in the body of the rule. Additionally, all constants in P should be DL-individuals. 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 rule-based reasoning is supported only for non DL-atoms, and DL-atoms in the body of the rules mainly express constraints.

In a subsequent work, Rosati (2005) defines the r-hybrid knowledge bases. In r-hybrid knowledge bases, DL-atoms are allowed in the head of the rules and the DL component T is a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base. Additionally, constants in P are not necessarily DL-individuals. However, a stronger safety condition is imposed, as each rule variable should appear in a (positive) non DL-atom in the body of the rule. Additionally, weak negation is allowed only for non DL-atoms and rule-based meta-reasoning is not supported. In general, we can say that for non DL-atoms, a closed-world assumption is made, while DL-atoms conform to the open-world assumption, as $\mathcal{SHOIN}(\mathbf{D})$ is a fragment of first-order logic.

11. Conclusions

In this paper, we have extended RDF graphs to ERDF graphs by allowing negative triples for representing explicit negative information. Then, we proceeded by defining an ERDF ontology as an ERDF graph complemented by a set of derivation rules with all connectives $\sim$ (weak negation), $\neg$ (strong negation), $\supset$ (material implication), $\wedge$, $\vee$, $\forall$, $\exists$ in the body of a rule, and with strong negation $\neg$ in the head of a rule. Moreover, we have extended the RDF(S) vocabulary by adding the predefined vocabulary elements *erdf*:*TotalProperty* and *erdf*:*TotalClass*, for representing the metaclasses of total properties and total classes, on which the open-world assumption applies.

We have defined ERDF formulas, ERDF interpretations, and ERDF entailment on ERDF formulas, showing that it conservatively 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 and, in particular, on its generalized definition of stable models (Herre & Wagner, 1997; Herre et al., 1999) (which extends 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 (a) the totality of properties and classes to be declared on a selective basis and (b) the explicit representation of closed-world assumptions (as derivation rules) enables the combination of open-world and closed-world reasoning in the same framework.

In particular, for a total property p, the open-world assumption applies, since each considered Herbrand interpretation I, in the computation of ERDF stable models, satisfies $p(x,y) \vee \neg p(x,y)$, for each pair (x,y) of ontology vocabulary terms. For a closed property p, a default closure rule of the form $\neg p(?x, ?y) \leftarrow \sim p(?x, ?y)$ is added, which allows to infer the falsity of $p(x,y)$, if there is no evidence that $p(x,y)$ holds. However, this method only works for partial properties. For a total property p, it may happen that there is a stable model, where $p(x,y)$ holds, even though there is no evidence for it (see the example in Section 5, above Proposition 5.1). In fact, if p is a total property, the existence or not of the corresponding default closure rule does not affect the ontology semantics.

The main advantages of ERDF are summarized as follows:

- It has a Tarski-style model theory, which is a desirable feature for logic languages for the Semantic Web (Bry & Marchiori, 2005).
- It is based on Partial Logic (Herre et al., 1999), which is the simplest conservative extension of classical logic that supports both weak and strong negation. Partial logic also extends Answer Set Programming (ASP)[22] (Gelfond & Lifschitz, 1990), by allowing all logical factors $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$, $\forall$, $\exists$ in the body of a rule.
- It enables the combination of open-world (monotonic) and closed-world (non-monotonic) reasoning, in the same framework.
- It extends RDFS ontologies with derivation rules and integrity constraints.

Satisfiability and entailment under the ERDF stable model semantics are in general undecidable. In a subsequent paper, we plan to identify syntactic restrictions for the ERDF ontologies that guarantee decidability of reasoning and to elaborate on the ERDF computability and complexity issues.

In this work, we consider only coherent ERDF interpretations. However, due to the Semantic Web's decentralized and distributed nature, contradictory information is frequent (Schaffert, Bry, Besnard, Decker, Decker, Enguix, & Herzig, 2005). Though Partial Logic allows for truth-value clashes, handling inconsistency in the Semantic Web is a topic that deserves extended treatment, which is outside the scope of this paper. It is in our future plans to consider general ERDF interpretations and extend the vocabulary of ERDF with the terms *erdf*:*CoherentProperty* and *erdf*:*CoherentClass*, whose instances are properties and classes that satisfy *coherence*. Thus, coherence will be decided on a per property and

22. ASP is a well-known and accepted knowledge representation formalism that allows (through credulous reasoning) the definition of concepts ranging over a space of choices. This feature enables the compact representation of search and optimization problems (Eiter, Ianni, Polleres, & Schindlauer, 2006).

per class basis. Admitting incoherent models will only be interesting in combination with a second preference criterion of "minimal incoherence" (Herre et al., 1999).

Our future work also concerns the support of datatypes, including XSD datatypes, and the extension of the predefined ERDF vocabulary by adding other useful constructs, possibly in accordance with the extensions of ter Horst (2005b). We also plan to formally define the ERDF/XML syntax, briefly presented in Section 7. Moreover, we plan to implement an ERDF inference engine.

Finally, we would like to mention that the success of the Semantic Web is impossible without support for modularity, encapsulation, information hiding, and access control. Modularity mechanisms and syntactic restrictions for merging knowledge bases in the Semantic Web are explored by Damásio et al. (2006). However, in this work, knowledge bases are expressed by extended logic programs. Our future plans include the extension of ERDF with mechanisms allowing sharing of knowledge between different ERDF ontologies, along the lines proposed by Damásio et al. (2006).

Acknowledgments

The authors would like to thank the reviewers for their valuable comments. This research has been partially funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE num. 506779 (www.rewerse.net).

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 the W3C Recommendation of RDF semantics (Hayes, 2004).

Let $\mathcal{URI}$ denote the set of URI references, $\mathcal{PL}$ denote the set of plain literals, and $\mathcal{TL}$ denote the set of typed literals, respectively. A vocabulary V is a subset of $\mathcal{URI} \cup \mathcal{PL} \cup \mathcal{TL}$. The vocabulary of RDF, $\mathcal{V}_{RDF}$, and the vocabulary of RDFS, $\mathcal{V}_{RDFS}$, are shown in Table 1 (Section 3).

Definition A.1 (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 $I_V\colon V \cap \mathcal{URI} \to Res_I \cup Prop_I$.
- A property extension mapping $PT_I : Prop_I \to \mathcal{P}(Res_I \times Res_I)$.
- 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: $I : V \to Res_I \cup Prop_I$ such that:

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

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

Definition A.2 (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 $[I+v](x) = I(x)$. We define:

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

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

Table 2: The RDF axiomatic triples

Definition A.3 (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
   $\langle IL_I($"s"^^$rdf{:}XMLLiteral), I(rdf{:}XMLLiteral) \rangle \in PT_I(I(rdf{:}type))$.
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. $\Box$

Definition A.4 (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 $I \models G'$. $\Box$

Definition A.5 (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:

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

Table 3: The RDFS axiomatic triples

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

Definition A.6 (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 lemmas and propositions presented in the main paper. In addition, we provide Lemma B.1, which is used in some of the proofs. To reduce the size of the proofs, we have eliminated the namespace from the URIs in $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$.

Lemma B.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 3.4, 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 a mapping $u_1 : Var(G) \rightarrow Res_I$ s.t. $u_1(y) = u(y)$, $\forall y \in Var(G) - \{x\}$, and $I, u_1 \models G$. Let u_2 be the mapping $u_2 : Var(G) \rightarrow Res_I$ s.t. $u_2(y) = u'(y)$, $\forall y \in Var(G) - \{x\}$, 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 $I, u' \models F$. $\square$

Lemma 3.1. Let G be an ERDF graph and let I be a partial interpretation of a vocabulary V. It holds: $I \models_{\texttt{GRAPH}} G$ iff $I \models formula(G)$.

Proof: Let $G = \{t_1, ..., t_n\}$ and $F = formula(G)$.

$\Rightarrow$) Assume that $I \models_{\texttt{GRAPH}} G$, we will show that $I \models F$. Since $I \models_{\texttt{GRAPH}} G$, 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 Lemma B.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 $I \models_{\texttt{GRAPH}} G$. 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, $I \models_{\texttt{GRAPH}} G$. $\square$

Proposition 3.1. 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 $TCls_I$. $\square$

Proposition 3.2. 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:
$\Leftarrow$) 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 $\exists v : Var(G) \rightarrow Res_I$ 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 $res : \mathcal{V}_{ERDF} \rightarrow R$, where R is a set disjoint from Res_I. Additionally, let $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Based on I and the mapping res, we construct a partial interpretation J of V as follows:

- $Res_J = Res_I \cup res(\mathcal{V}_{ERDF})$.
- $J_V(x) = I_V(x),\ \forall x \in (V' - \mathcal{V}_{ERDF}) \cap \mathcal{URI}$ and $J_V(x) = res(x),\ \forall x \in \mathcal{V}_{ERDF}$.
- We define the mapping: $IL_J : V' \cap \mathcal{TL} \rightarrow Res_J$ such that: $IL_J(x) = IL_I(x)$.
- We define the mapping: $J : V' \rightarrow Res_J$ such that:
  - $J(x) = J_V(x),\ \forall x \in V' \cap \mathcal{URI}$.
  - $J(x) = x,\ \forall\, x \in V' \cap \mathcal{PL}$.
  - $J(x) = IL_J(x),\ \forall\, x \in V' \cap \mathcal{TL}$.
- We define the mapping $PT'_J : Res_J \rightarrow \mathcal{P}(Res_J \times Res_J)$ 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) $\langle res(\mathit{TotalClass}), J(\mathit{Class})\rangle \in PT'_J(J(\mathit{subClassOf}))$.
  (PT3) $\langle res(\mathit{TotalProperty}), J(\mathit{Property})\rangle \in PT'_J(J(\mathit{subClassOf}))$.

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

  (PT4) if $\langle x, y\rangle \in PT'_J(J(\mathit{domain}))$ and $\langle z, w\rangle \in PT'_J(x)$ then $\langle z, y\rangle \in PT'_J(J(\mathit{type}))$.
  (PT5) if $\langle x, y\rangle \in PT'_J(J(\mathit{range}))$ and $\langle z, w\rangle \in PT'_J(x)$ then $\langle w, y\rangle \in PT'_J(J(\mathit{type}))$.
  (PT6) if $\langle x, J(\mathit{Class})\rangle \in PT'_J(J(\mathit{type}))$ then $\langle x, J(\mathit{Resource})\rangle \in PT'_J(J(\mathit{subClassOf}))$.
  (PT7) if $\langle x, y\rangle \in PT'_J(J(\mathit{subClassOf}))$ then $\langle x, J(\mathit{Class})\rangle \in PT'_J(J(\mathit{type}))$.
  (PT8) if $\langle x, y\rangle \in PT'_J(J(\mathit{subClassOf}))$ then $\langle y, J(\mathit{Class})\rangle \in PT'_J(J(\mathit{type}))$.
  (PT9) if $\langle x, y\rangle \in PT'_J(J(\mathit{subClassOf}))$ and $\langle z, x\rangle \in PT'_J(J(\mathit{type}))$ then $\langle z, y\rangle \in PT'_J(J(\mathit{type}))$.
  (PT10) if $\langle x, J(\mathit{Class})\rangle \in PT'_J(J(\mathit{type}))$ then $\langle x, x\rangle \in PT'_J(J(\mathit{subClassOf}))$.
  (PT11) if $\langle x, y\rangle \in PT'_J(J(\mathit{subClassOf}))$ and $\langle y, z\rangle \in PT'_J(J(\mathit{subClassOf}))$ then $\langle x, z\rangle \in PT'_J(J(\mathit{subClassOf}))$.
  (PT12) if $\langle x, y\rangle \in PT'_J(J(\mathit{subPropertyOf}))$ then $\langle x, J(\mathit{Property})\rangle \in PT'_J(J(\mathit{type}))$.
  (PT13) if $\langle x, y\rangle \in PT'_J(J(\mathit{subPropertyOf}))$ then $\langle y, J(\mathit{Property})\rangle \in PT'_J(J(\mathit{type}))$.
  (PT14) if $\langle x, y\rangle \in PT'_J(J(\mathit{subPropertyOf}))$ and $\langle z, w\rangle \in PT'_J(x)$ then $\langle z, w\rangle \in PT'_J(y)$.

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

After reaching fixpoint, nothing else is contained in $PT'_J(x)$, $\forall x \in \mathit{Res}_J$.

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

(PF1) if "s"^^$\mathit{rdf{:}XMLLiteral} \in V$ is an ill-typed XML-Literal then $\langle IL_J(\text{"}s\text{"\^\^}\mathit{rdf{:}XMLLiteral}), J(\mathit{Literal})\rangle \in PF_J(J(\mathit{type}))$.
(PF2) if $\langle J(\mathit{TotalClass}), J(\mathit{TotalClass})\rangle \in PT_J(J(\mathit{type}))$ then $\forall x \in \mathit{Res}_J - \{J(\mathit{TotalClass})\},\ \ \langle x, J(\mathit{TotalClass})\rangle \in PF_J(J(\mathit{type}))$.
(PF3) if $\langle J(\mathit{TotalProperty}), J(\mathit{TotalProperty})\rangle \in PT_J(J(\mathit{type}))$ then $\forall x, y \in \mathit{Res}_J,\ \ \langle x, y\rangle \in PF_J(J(\mathit{TotalProperty}))$.

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

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

After reaching fixpoint, nothing else is contained in $PF_J(x)$, $\forall x \in \mathit{Prop}_J$.

Before we continue, we prove the following lemma:

Lemma: For all $x, y, x \in \mathit{Res}_J$, $\ \langle x, y\rangle \in PT'_J(z)$ 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) $z \in \mathit{Prop}_I$ or (ii) $\exists w \in \mathit{Res}_J$, s.t. $\langle w, z\rangle \in PT'_J(J(\mathit{subPropertyOf}))$.

(i) Assume that $z \in \mathit{Prop}_I$. Then, $\langle z, I(\mathit{Property})\rangle \in PT_I(I(\mathit{type}))$. This implies that $\langle z, J(\mathit{Property})\rangle \in PT_I(J(\mathit{type}))$. From (PT1), it now follows that $\langle z, J(v)\rangle \in PT'_J(J(\mathit{type}))$. Therefore, $z \in \mathit{Prop}_J$. 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 $\langle z, J(Property) \rangle \in PT'_J(J(type))$. Therefore, $z \in Prop_J$. From the definition of PT_J, it now follows that $\langle x, y \rangle \in PT_J(z)$.
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 $CT_J, CF_J : Cls_J \rightarrow \mathcal{P}(Res_J)$ 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 3.7 (ERDF interpretation) and Definition 3.2 (Coherent ERDF interpretation).

First, we will show that J satisfies semantic condition 2 of Definition 3.7. 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, it holds that $\langle x, I(Resource) \rangle \in PT_I(I(type))$. Thus, it holds $\langle x, J(Resource) \rangle \in PT_J(J(type))$, 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 3.7. 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 3.7. 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 3.7. 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 3.7. 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 3.7.

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:

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

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 $\langle x, y\rangle \in PT_I(z)$[23].
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 3.7. 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 3.7. 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 3.7.
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: $\langle IL_I(``s\text{''}\hat{}\hat{}rdf{:}XMLLiteral), I(XMLLiteral)\rangle \in PT_I(I(type))$. Therefore, from (PT1), it follows that $\langle IL_J(``s\text{''}\hat{}\hat{}rdf{:}XMLLiteral), J(XMLLiteral)\rangle \in PT_J(J(type))$.

We will now show that J satisfies semantic condition 15 of Definition 3.7. Let "s"^^rdf:$XMLLiteral$ $\in V$ s.t. s is not a well-typed XML literal string. Assume now that IL_J("s"^^rdf:$XMLLiteral$) $\in LV_J$. Then, $\langle IL_J(``s\text{''}\hat{}\hat{}rdf{:}XMLLiteral), J(Literal)\rangle \in PT_J(J(type))$. From *Observation 4*, it follows that $\langle IL_J(``s\text{''}\hat{}\hat{}rdf{:}XMLLiteral), J(Literal)\rangle \in PT_I(J(type))$. Therefore, it follows that $\langle IL_I(``s\text{''}\hat{}\hat{}rdf{:}XMLLiteral), I(Literal)\rangle \in$

23. Note that *Observation 3* implies *Observation 4*.

$PT_I(I(type))$. Thus, $IL_I(\text{“}s\text{”\^\^}rdf{:}XMLLiteral) \in LV_I$, which is impossible since I is an RDFS interpretation of V. Therefore, $IL_J(\text{“}s\text{”\^\^}rdf{:}XMLLiteral) \in Res_J - LV_J$.

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 16 of Definition 3.7, due to (PT1). Finally, J satisfies semantic condition 17, 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 3.2). 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 $\langle x, y\rangle \in PF_J(J(type))$.
Now, from *Observation 1* and since $z \in Res_I$, (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 $\langle x, y\rangle \in PF_J(J(type))$.
Now, from *Observation 1* and since $z \in Res_I$, (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 $p \in V'$, $s, o \in V' \cup Var$. 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 $u : Var(G') \rightarrow Res_J = Res_I \cup res(\mathcal{V}_{ERDF})$ s.t. $J, u \models G'$. We define a mapping $u' : Var(G') \rightarrow Res_I$ as follows:

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

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 $p \in V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS}$, $s, o \in V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup Var$, 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 $u(s) = res(TotalProperty)$, 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 $\langle I(Class), I(v)\rangle \in PT_I(z)$. 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 $u(o) = res(TotalProperty)$, 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: $Res_J = Res_I$, $Prop_J = Prop_I$, $LV_J = LV_I$, $Cls_J = Cls_I$, $J_V(x) = I_V(x), \forall x \in V' \cap \mathcal{URI}$, $PT_J(x) = PT_I(x), \forall x \in Prop_J$, $IL_J(x) = IL_I(x), \forall x \in V' \cap \mathcal{TL}$, $CT_J(x) = CT_I(x), \forall x \in Cls_J$.

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 A.3 (Appendix A, 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 now show that J satisfies semantic condition 2 of Definition A.3. 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 $IL_J(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral)$ is the XML value of s, and $IL_J(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral) \in CT_J(J(XMLLiteral))$. We will show that $IL_J(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral) \in LV_J$. Since I is an ERDF interpretation, $IL_I(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral) \in CT_I(I(XMLLiteral))$. Additionally, $\langle I(XMLLiteral), I(Literal)\rangle \in PT_I(I(subClassOf))$. Therefore, $IL_I(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral) \in CT_I(I(Literal))$, and thus, $IL_I(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral) \in LV_I$. The last statement implies that $IL_J(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral) \in LV_J$.

We will now show that J satisfies semantic condition 3 of Definition A.3. 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 $IL_J(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral) \in Res_J - LV_J$. We will show that
$\langle IL_J(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral), J(XMLLiteral)\rangle \notin PT_J(J(type))$. Assume that
$\langle IL_J(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral), J(XMLLiteral)\rangle \in PT_J(J(type))$. Then,
$\langle IL_I(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral), I(XMLLiteral)\rangle \in PT_I(I(type))$. Thus,
$IL_I(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral) \in CT_I(I(XMLLiteral))$. Since it holds
$\langle I(XMLLiteral), I(Literal)\rangle \in PT_I(I(subClassOf))$, it follows that
$IL_I(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral) \in CT_I(I(Literal))$. Thus, $IL_I(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral) \in LV_I$, which is impossible since I is an ERDF interpretation of V. Therefore,
$\langle IL_J(\text{"}s\text{"}\hat{}\hat{}rdf{:}XMLLiteral), J(XMLLiteral)\rangle \notin PT_J(J(type))$.

It is easy to see that J satisfies semantic condition 4 of Definition A.3 and all the semantic conditions of Definition A.5 (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 $I \models G'$. $\Box$

Proposition 4.1. Let G be an ERDF graph and let 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. $I \models sk(G)$. We will show that $I \models G$. 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.

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 $\langle I(u'(s)), I(u'(o))\rangle \in PT_I(I(p))$, which is true, since $p(u'(s), u'(o)) \in sk(G)$ and $I \models sk(G)$. Thus, $I, u \models p(s, o)$.

Let $\neg p(s,o) \in G$. Then, $p \in V'$, $s,o \in V' \cup \mathit{Var}$, and $I(p) \in \mathit{Prop}_I$. It holds: $\langle [I+u](s), [I+u](o) \rangle \in \mathit{PF}_I(I(p))$ iff $\langle I(u'(s)), I(u'(o)) \rangle \in \mathit{PF}_I(I(p))$, which is true, since $\neg p(u'(s), u'(o)) \in \mathit{sk}(G)$ and $I \models \mathit{sk}(G)$. Thus, $I, u \models \neg p(s,o)$.

Therefore, $I \models G$. Since $G \models^{ERDF} F$, it follows that $I \models F$.

$\Leftarrow$) Let $\mathit{sk}(G) \models^{ERDF} F$. 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 $I \models F$. Since $I \models G$, there is a total function $u : \mathit{Var}(G) \rightarrow \mathit{Res}_I$ 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 $V \cup \mathit{sk}_G(\mathit{Var}(G))$ as follows: $\mathit{Res}_J = \mathit{Res}_I$, $\mathit{Prop}_J = \mathit{Prop}_I$, $LV_J = LV_I$, $\mathit{Cls}_J = \mathit{Cls}_I$. We define $J_V : (V' \cup \mathit{sk}_G(\mathit{Var}(G))) \cap \mathcal{URI} \rightarrow \mathit{Res}_J$, as follows: $J_V(x) = I_V(x), \forall x \in V' \cap \mathcal{URI}$ and $J_V(x) = u(\mathit{sk}_G^{-1}(x)), \forall x \in \mathit{sk}_G(\mathit{Var}(G))$. Moreover, $PT_J(x) = PT_I(x), \forall x \in \mathit{Prop}_J$, $PF_J(x) = PF_I(x), \forall x \in \mathit{Prop}_J$, $IL_J(x) = IL_I(x), \forall x \in V' \cap \mathcal{TL}$, $CT_J(x) = CT_I(x), \forall x \in \mathit{Cls}_J$, and $CF_J(x) = CF_I(x), \forall x \in \mathit{Cls}_J$.

Since I is an ERDF interpretation of V, it is easy to see that J is indeed an ERDF interpretation of $V \cup \mathit{sk}_G(\mathit{Var}(G))$. We will show that $J \models \mathit{sk}(G)$. First, we define a total function $g : V' \cup \mathit{sk}_G(\mathit{Var}(G)) \rightarrow V' \cup \mathit{Var}(G)$ as follows: $g(x) = \mathit{sk}_G^{-1}(x)$, $\forall x \in \mathit{sk}_G(\mathit{Var}(G))$ and $g(x) = x$, otherwise. Let $p(s,o) \in \mathit{sk}(G)$. Since $I \models G$, it follows that $p \in V'$, $s,o \in V' \cup \mathit{Var}$, and $J(p) = I(p) \in \mathit{Prop}_I = \mathit{Prop}_J$. It holds $J(s) = [I+u](g(s))$, $J(o) = [I+u](g(o))$, 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 $I, u \models G$. Let $v : \{\} \rightarrow \mathit{Res}_J$. It follows that $J, v \models p(s,o)$. Let $\neg p(s,o) \in \mathit{sk}(G)$. We can show that $J, v \models \neg p(s,o)$, in a similar manner. Therefore, $J \models \mathit{sk}(G)$.

Since $\mathit{sk}(G) \models^{ERDF} F$, it follows that $J \models F$. We will show that $I \models F$. We define $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Note that $\mathit{Res}_J = \mathit{Res}_I$.

Lemma: For every mapping $u : \mathit{Var}(F) \rightarrow \mathit{Res}_J$, 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 3.4, to get an equivalent formula that satisfies the assumption.

Let $F = p(s,o)$. Assume that $J, u \models F$. Since $V_F \cap \mathit{sk}_G(\mathit{Var}(G)) = \emptyset$, it follows that $p \in V'$, $s,o \in V' \cup \mathit{Var}$, and $J(p) = I(p) \in \mathit{Prop}_I = \mathit{Prop}_J$. 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 $p \in V'$, $s,o \in V' \cup \mathit{Var}$, and $J(p) = I(p) \in \mathit{Prop}_I = \mathit{Prop}_J$. 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 $v : Var(G) \rightarrow Res_I$ s.t. $v(y) = u(y),\ \forall y \in Var(G) - \{x\}$ and $I, v \models G$ iff there is $v : Var(G) \rightarrow Res_J$ 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 $u : Var(F) \rightarrow Res_J, \quad J, u \models F$. Therefore, it follows from Lemma and the fact that $Res_J = Res_I$ that for every mapping $u : Var(F) \rightarrow Res_I, \quad I, u \models F$. Thus, $I \models F$. $\Box$

Proposition 4.2. Let $O = \langle G, P \rangle$ be an ERDF ontology and let $I, J \in \mathcal{I}^H(O)$. Let $p \in TProp_I \cap TProp_J$. If $PT_I(p) \neq PT_J(p)$ or $PF_I(p) \neq PF_J(p)$ then $I \not\leq J$ and $J \not\leq I$.
Proof: Assume $PT_I(p) \neq PT_J(p)$. Now, assume $I \leq J$. Then, $PT_I(p) \subset PT_J(p)$ and $PF_I(p) \subseteq PF_J(p)$. Since $I, J \in \mathcal{I}^H(O)$ and $p \in TProp_I \cap TProp_J$, it holds that $PF_I(p) = Res^H_O - PT_I(p)$ and $PF_J(p) = Res^H_O - PT_J(p)$. Thus, $PF_I(p) \supset PF_J(p)$, which is a contradiction. Thus, $I \not\leq J$. Similarly, we can prove that $J \not\leq I$.

Assume now that $PF_I(p) \neq PF_J(p)$. Then, we can prove that $I \not\leq J$ and $J \not\leq I$, in a similar manner. $\Box$

Proposition 5.1. 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 $v : Var(r) \rightarrow Res^H_O$ s.t. $M, v \models Cond(r)$. It is enough to show that $M, v \models Concl(r)$.

For any mapping $u : X \rightarrow Res^H(O)$, where $X \subseteq Var$, we define the mapping $u^* : X \rightarrow V_O$ as follows:

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

Let $x \in V_O$, we define $x^{u^*} = x$. Let $x \in X$, we define $x^{u^*} = u^*(x)$. Let $F \in L(V_O) \cup \{true, false\}$ such that $FVar(F) \subseteq X$, we define F^{u^*} to be the formula that results from F after replacing each free variable of F by $u^*(x)$. 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 an ERDF formula over V_O and let u be a mapping $u : Var(F) \rightarrow Res^H_O$. It holds: $M, u \models F$ iff $M, u \models F^{u^*}$.
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 3.4, 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 $\langle [M + u](s^{u^*}), [M + u](o^{u^*}) \rangle \in PT_M(M(p))$ iff $M, u \models p(s, o)^{u^*}$.

Let $F = \neg p(s, o)$. It holds: $M, u \models F$ iff $M, u \models p(s, o)$ iff $\langle [M + u](s), [M + u](o) \rangle \in PF_M(M(p))$ iff $\langle [M + u](s^{u^*}), [M + u](o^{u^*}) \rangle \in PF_M(M(p))$ iff $M, u \models (\neg p(s, o))^{u^*}$.
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 $M, u \not\models G^{u^*}$ iff $M, u \models {\sim}G^{u^*}$ iff $M, u \models F^{u^*}$.

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 $M, u \models F_1^{u^*}$ and $M, u \models F_2^{u^*}$ iff $M, u \models (F_1 \wedge F_2)^{u^*}$ iff $M, u \models F^{u^*}$.

Let $F = \exists x G$. It holds: $M, u \models F$ iff there exists a mapping $u_1 : \mathit{Var}(G) \to \mathit{Res}_O^H$ s.t. $u_1(y) = u(y),\ \forall y \in \mathit{Var}(G) - \{x\}$ s.t. $M, u_1 \models G$ iff there exists a mapping $u_1 : \mathit{Var}(G) \to \mathit{Res}_O^H$ s.t. $u_1(y) = u(y),\ \forall y \in \mathit{Var}(G) - \{x\}$ s.t. $M, u_1 \models G^{u_1^*}$ iff there exists a mapping $u_1 : \mathit{Var}(G) \to \mathit{Res}_O^H$ s.t. $u_1(y) = u(y),\ \forall y \in \mathit{Var}(G) - \{x\}$ s.t. $M, u_1 \models (\exists x G)^{u_1^*}$ iff (since $u_1^*(y) = u^*(y),\ \forall y \in \mathit{FVar}(\exists x G))$ $M, u \models (\exists x G)^{u^*}$ iff $M, u \models F^{u^*}$.

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 $M, u \models F$ iff $M, u \models F^{u^*}$.
End of Lemma

First assume that $\mathit{Cond}(r) \neq \mathit{true}$. Then, $\mathit{Cond}(r) \in L(V_O)$ and thus, $\mathit{Cond}(r)$ is an ERDF formula over V_O. Since $M, v \models \mathit{Cond}(r)$, it follows from Lemma that $M, v \models \mathit{Cond}(r)^{v^*}$. Now since $\mathit{FVar}(\mathit{Cond}(r)^{v^*}) = \emptyset$, it follows from Lemma B.1 that $M \models \mathit{Cond}(r)^{v^*}$. Since $M \in \mathcal{M}^{st}(O)$, it follows that $M \models \mathit{Concl}(r)^{v^*}$. Thus, $\mathit{Concl}(r) \neq \mathit{false}$ and $\mathit{Concl}(r) \in L(V_O|\{\neg\})$. Now since $\mathit{FVar}(\mathit{Concl}(r)^{v^*}) = \emptyset$, it follows from lemma B.1 that $M, v \models \mathit{Concl}(r)^{v^*}$. Since $\mathit{Concl}(r)$ is an ERDF formula over V_O, it follows from Lemma that $M, v \models \mathit{Concl}(r)$.

Assume now that $\mathit{Cond}(r) = \mathit{true}$. Then, $M \models \mathit{Cond}(r)^{v^*}$. Since $M \in \mathcal{M}^{st}(O)$, it follows that $M \models \mathit{Concl}(r)^{v^*}$. Therefore, $\mathit{Concl}(r) \neq \mathit{false}$, and we can prove as above that $M, v \models \mathit{Concl}(r)$.

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

Proposition 5.2. Let $O = \langle G, P \rangle$ be an ERDF ontology, such that $\mathit{rdfs{:}subClassOf}(\mathit{rdf{:}Property}, \mathit{erdf{:}TotalProperty}) \in G$. Then, $\mathcal{M}^{st}(O) = \mathcal{M}^H(O)$.

Proof: From Proposition 5.1, 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 $M \in \mathit{minimal}(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})$.

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 $\mathit{Prop}_J \subseteq \mathit{Prop}_M$ and for all $p \in \mathit{Prop}_J$, it holds $\mathit{PT}_J(p) \subseteq \mathit{PT}_M(p)$ and $\mathit{PF}_J(p) \subseteq \mathit{PF}_M(p)$. Let $p \in \mathit{Prop}_J$. Since $J \models sk(G)$, it follows that $\mathit{Prop}_J \subseteq \mathit{TProp}_J$. Thus, $p \in \mathit{TProp}_J$. Assume that $\mathit{PT}_J(p) \neq \mathit{PT}_M(p)$. Then, there is $\langle x, y \rangle \in \mathit{PT}_M(p)$ s.t. $\langle x, y \rangle \notin \mathit{PT}_J(p)$. Then, $\langle x, y \rangle \in \mathit{PF}_J(p)$. Thus, $\langle x, y \rangle \in \mathit{PF}_M(p)$, which is impossible, since $\langle x, y \rangle \in \mathit{PT}_M(p)$. Thus, $\mathit{PT}_J(p) = \mathit{PT}_M(p)$. Similarly, we can prove that $\mathit{PF}_J(p) = \mathit{PF}_M(p)$. Therefore, for all $p \in \mathit{Prop}_J$, it holds $\mathit{PT}_J(p) = \mathit{PT}_M(p)$ and $\mathit{PF}_J(p) = \mathit{PF}_M(p)$. We will now show that $\mathit{Prop}_J = \mathit{Prop}_M$. It holds $\mathit{Prop}_J = \{x \in \mathit{Res}_O^H \mid \langle x, \mathit{Property} \rangle \in \mathit{PT}_J(\mathit{type})\} = \{x \in \mathit{Res}_O^H \mid \langle x, \mathit{Property} \rangle \in \mathit{PT}_I(\mathit{type})\} = \mathit{Prop}_M$. Based on these results, the fact that $J, M \in \mathcal{I}^H(O)$, it follows that $J = M$. Therefore, $M \in \mathit{minimal}(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})$.

We will now show that $M \in \mathit{minimal}(\{I \in \mathcal{I}^H(O) \mid I \geq M$ and $I \models \mathit{Concl}(r)$, for all $r \in P_{[M,M]}\})$. Since $M \in \mathcal{M}^H(O)$ it follows that $M \in \{I \in \mathcal{I}^H(O) \mid I \geq M$ and $I \models \mathit{Concl}(r)$, for all $r \in P_{[M,M]}\}$. Let $J \in \{I \in \mathcal{I}^H(O) \mid I \geq M$ and $I \models \mathit{Concl}(r)$, for all $r \in P_{[M,M]}\}$ and $J \leq M$. Since $J \geq M$, it follows that $\mathit{Prop}_M \subseteq \mathit{Prop}_J$, and for all $p \in \mathit{Prop}_M$, it holds $\mathit{PT}_M(p) \subseteq \mathit{PT}_J(p)$ and $\mathit{PF}_M(p) \subseteq \mathit{PF}_J(p)$. 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, the fact that $J, M \in \mathcal{I}^H(O)$, it follows that $J = M$.

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

Since M satisfies the conditions of Definition 5.1 (Stable Model), it follows that $M \in \mathcal{M}^{st}(O)$. Thus, it holds $\mathcal{M}^H(O) \subseteq \mathcal{M}^{st}(O)$.

Therefore, $\mathcal{M}^H(O) = \mathcal{M}^{st}(O)$. $\square$

Proposition 6.2. Let G be an ERDF graph and let F be an ERDF formula such that $V_F \cap sk_G(Var(G)) = \emptyset$. It holds:

1. If F is an ERDF *d*-formula and $\langle G, \emptyset \rangle \models^{st} F$ then $G \models^{ERDF} F$.
2. If $G \models^{ERDF} F$ then $\langle G, \emptyset \rangle \models^{st} F$.

Proof:
1) Let $\langle G, \emptyset \rangle \models^{st} F$. We will show that $sk(G) \models^{ERDF} F$. Let I be an ERDF interpretation of a vocabulary V s.t. $I \models sk(G)$. We will show that $I \models F$. 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:

- $Res_J = Res_O^H$.
- $J_V(x) = x$, for all $x \in V_O \cap \mathcal{URI}$.
- 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:
  - $J(x) = J_V(x),\ \forall x \in V_O \cap \mathcal{URI}$.
  - $J(x) = x,\ \forall\, x \in V_O \cap \mathcal{PL}$.
  - $J(x) = IL_J(x),\ \forall\, x \in V_O \cap \mathcal{TL}$.
- $Prop_J = \{x \in Res_J \mid \exists x' \in V_O,\ J(x') = x$ 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, $\langle x, I(\mathit{Literal})\rangle \in PT_I(I(\mathit{type}))$. This implies that $\langle x, J(\mathit{Literal})\rangle \in PT_J(J(\mathit{type}))$. Thus, $x \in LV_J$.

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

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 3.7 (ERDF Interpretation), in a number of steps:
Step 1: Here, we prove that $\mathit{Res}_J = CT_J(J(\mathit{Resource}))$. Obviously, $CT_J(J(\mathit{Resource})) \subseteq \mathit{Res}_J$. We will show that $\mathit{Res}_J \subseteq CT_J(J(\mathit{Resource}))$. Let $x \in \mathit{Res}_J$. Then, there is $x' \in V_O$ such that $J(x') = x$. We want to show that $\langle J(x'), J(\mathit{Resource})\rangle \in PT_J(J(\mathit{type}))$. It holds: $\langle J(x'), J(\mathit{Resource})\rangle \in PT_J(J(\mathit{type}))$ iff $\langle I(x'), I(\mathit{Resource})\rangle \in PT_I(I(\mathit{type}))$, which is true, since I is an ERDF interpretation that satisfies $sk(G)$ and $I(x') \in \mathit{Res}_I$. Thus, $x = J(x') \in CT_J(J(\mathit{ResourceResource}))$.
Therefore, $\mathit{Res}_J = CT_J(J(\mathit{Resource}))$.

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

We will now show that $CT_J(J(\mathit{Property})) \subseteq \mathit{Prop}_J$. Let $x \in CT_J(J(\mathit{Property}))$. Then, $\exists x' \in V_O$ such that $J(x') = x$. It holds $\langle J(x'), J(\mathit{Property})\rangle \in PT_J(J(\mathit{type}))$, which implies that $\langle I(x'), I(\mathit{Property})\rangle \in PT_I(I(\mathit{type}))$. Thus, $I(x') \in \mathit{Prop}_I$ and $x \in \mathit{Prop}_J$.
Therefore, $CT_J(J(\mathit{Property})) \subseteq \mathit{Prop}_J$.

Step 3: By definition, it holds $\mathit{Cls}_J = CT_J(J(\mathit{Class}))$, $LV_J = CT_J(J(\mathit{Literal}))$, $\mathit{TCls}_J = CT_J(J(\mathit{TotalClass}))$ and $\mathit{TProp}_J = CT_J(J(\mathit{TotalProperty}))$.

We will now show that J satisfies semantic condition 3 of Definition 3.7 (ERDF Interpretation). Let $\langle x, y\rangle \in PT_J(J(\mathit{domain}))$ 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, $\langle J(x'), J(y')\rangle \in PT_J(J(\mathit{domain}))$. 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, $\langle I(x'), I(y')\rangle \in PT_I(I(\mathit{domain}))$ and $\langle I(z'), I(w')\rangle \in PT_I(I(x'))$. Since I is an ERDF interpretation, $\langle I(z'), I(y')\rangle \in PT_I(I(\mathit{type}))$. Thus, $\langle J(z'), J(y')\rangle \in PT_J(J(\mathit{type}))$ 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 3.7. Thus, J is an ERDF interpretation of V_O.

Moreover, we will show that J is a coherent ERDF interpretation (Definition 3.2). Assume that this is not the case. Thus, there is $z \in \mathit{Prop}_J$ s.t. $PT_J(z) \cap PF_J(z) \neq \emptyset$. Thus, there are $x, y \in \mathit{Res}_J$ 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 since $I \models sk(G)$. Let $u : \{\} \rightarrow Res^H_O$. Then, $J, u \models p(s, o)$. Let $\neg p(s, o) \in sk(G)$. We can show that $J, u \models \neg p(s, o)$, in a similar manner. Thus, $J \models sk(G)$.

Now, from Definition 5.1 (Stable Model) and the fact that $J \models sk(G)$, it follows that $\exists K \in \mathcal{M}^{st}(O)$ s.t. $K \leq J$. From this and the fact that $O \models^{st} F$, it follows that $K \models F$. Since F is an ERDF d-formula, it holds that

$$F = (\exists ?x_1, ..., \exists ?x_{k_1}\ F_1) \vee ... \vee (\exists ?x_1, ..., \exists ?x_{k_n}\ F_n),$$

where $F_i = t_1 \wedge ... \wedge t_{m_i}$ and t_j, for $j = 1, ..., m_i$, is an ERDF triple. Thus, there is an $i \in \{1, ..., n\}$ and $u : Var(F_i) \rightarrow Res^H_O$ s.t. $K, u \models F_i$.

We will show that $J, u \models F_i$.

Let $p(s, o) \in \{t_1, ..., t_{m_i}\}$. Since K is an ERDF interpretation of V_O, $K, u \models F_i$, 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 $\neg p(s, o) \in \{t_1, ..., t_{m_i}\}$. Since K is an ERDF interpretation of V_O, $K, u \models F_i$, 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 $u' : V_{F_i} \cup Var(F_i) \rightarrow V_O$, as follows:

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

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

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

Let $p(s, o) \in \{t_1, ..., t_{m_i}\}$. Then, $p \in V_{F_i}$ and $s, o \in V_{F_i} \cup Var$. Since $J, u \models F_i$, it follows that $V_{F_i} \subseteq V_O$. Therefore, $V_{F_i} \subseteq V_{sk(G)} \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF} \subseteq V'$. 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 $J, u \models F_i$.

We want to show that $\langle [I+v''](s), [I+v''](o)\rangle \in PT_I(I(p))$. Note that $\forall x \in V_{F_i}$, it holds: $[I+u''](x) = I(u'(x)) = I(x)$ and $J(u'(x)) = [J+u](x) = J(x)$. Moreover, $\forall x \in Var(F_i)$, it holds: $[I+u''](x) = I(u'(x))$ and $J(u'(x)) = [J+u](x)$ (recall the definition of $J(.)$). 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 $J, u \models F_i$. Thus, $I, u'' \models p(s,o)$.
Let $\neg p(s,o) \in \{t_1, ..., t_{m_i}\}$. We can show that $I, u'' \models \neg p(s,o)$, in a similar manner.
Thus, $I, u'' \models F_i$, which implies that $I, u'' \models \exists\ ?x_1, ..., ?x_{k_i}\ F_i$. Thus, $I, u'' \models F$. Now, it follows from Lemma B.1 that $I \models F$.

Thus, $sk(G) \models^{ERDF} F$. Now, it follows from Proposition 4.1 that $G \models^{ERDF} F$.

2) Let $G \models^{ERDF} F$. It follows from Proposition 4.1 that $sk(G) \models^{ERDF} F$. 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 8.1 Let $\mathcal{D}$ be an instance of the unbounded tiling problem. It holds:

1. $\mathcal{D}$ has a solution iff $O_{\mathcal{D}} \cup \{false \leftarrow F_{\mathcal{D}}\}$ has a stable model.

2. $\mathcal{D}$ has a solution iff $O_{\mathcal{D}} \not\models^{st} F_{\mathcal{D}}$.

Proof:
1) This statement follows easily from statement 2).

2) $\Rightarrow$) Let τ be a solution to $\mathcal{D}$. Since $I\!N \times I\!N$ is denumerable, there exists a bijective function $\pi : I\!N \times I\!N \rightarrow I\!N$. Consider now a Herbrand interpretation I of $O_{\mathcal{D}}$ such that:

1. $CT_I(Tile) = CT_I(HasRight) = CT_I(HasAbove) = \{rdf{:}\_i \mid i \in I\!N\}$ and $CF_I(Tile) = CF_I(HasRight) = CF_I(HasAbove) = \emptyset$.

2. $PT_I(id) = \{\langle x, x\rangle \mid x \in V_O\}$ and $PF_I(id) = \emptyset$.

3. $PT_I(HConstraint) = H$ and $PF_I(HConstraint) = \emptyset$.

4. $PT_I(VConstraint) = V$ and $PF_I(VConstraint) = \emptyset$.

5. $PT_I(Type) = \{\langle rdf{:}\_\pi(i,j), \tau(i,j)\rangle \mid i, j \in I\!N\}$ and $PF_I(Type) = \emptyset$.

6. $PT_I(right) = \{\langle rdf{:}\_\pi(i,j), rdf{:}\_\pi(i+1,j)\rangle \mid i, j \in I\!N\}$ and $PF_I(right) = \{\langle rdf{:}\_i, rdf{:}\_j\rangle \mid i, j \in I\!N$ and $\langle rdf{:}\_i, rdf{:}\_j\rangle \notin PT_I(right)\}$.

7. $PT_I(above) = \{\langle rdf{:}\_\pi(i,j), rdf{:}\_\pi(i,j+1)\rangle \mid i, j \in I\!N\}$ and $PF_I(above) = \{\langle rdf{:}\_i, rdf{:}\_j\rangle \mid i, j \in I\!N$ and $\langle rdf{:}\_i, rdf{:}\_j\rangle \notin PT_I(above)\}$.

It is easy to see that I is a stable model of $O_{\mathcal{D}}$ and $I \not\models F_{\mathcal{D}}$. Thus, $O_{\mathcal{D}} \not\models^{st} F_{\mathcal{D}}$.

$\Leftarrow$) Let $\mathcal{D} = \langle \mathcal{T}, H, V\rangle$, where $\mathcal{T} = \{T_1, ..., T_n\}$. Assume that $O_{\mathcal{D}} \not\models^{st} F_{\mathcal{D}}$ and let I be a stable model of $O_{\mathcal{D}} = \langle G, P\rangle$ such that $I \not\models F_{\mathcal{D}}$. Obviously, $CT_I(Tile) = \{rdf{:}\_i \mid i \in I\!N\}$. Due to rule sets (2)-(4) of P and since $O_{\mathcal{D}} \not\models^{st} F_{\mathcal{D}}$, it holds that starting from tile $rdf{:}\_0$

and placing tiles according to $PT_I(right)$ and $PT_I(above)$ relations, a grid is formed. We define $\pi(i, j) = k$, for $i, j, k \in I\!N$, iff the tile $rdf\!:\_k$ has been placed on the $\langle i, j\rangle$ position of the previous grid. Note that π is a total function. Due to rule set (1) of P, each tile is assigned a unique type in $\mathcal{T} = \{T_1, ..., T_n\}$. Due to rule set (5) of P, this type assignment satisfies the horizontal and vertical adjacency constraints of $\mathcal{D}$. Thus, a solution of $\mathcal{D}$ is $\tau : I\!N \times I\!N \rightarrow \mathcal{T}$, where $\tau(i, j) = T$ iff $\langle rdf\!:\_\pi(i, j), T\rangle \in PT_I(Type)$. Since π is a total function and, for all $k \in I\!N$, tile $rdf\!:\_k$ is assigned a unique type in $\mathcal{T}$, it follows that τ is a total function.

References

Alferes, J. J., Damásio, C. V., & Pereira, L. M. (1995). A Logic Programming System for Non-monotonic Reasoning. *Special Issue of the Journal of Automated Reasoning*, *14*(1), 93–147.

Alferes, J. J., Damásio, C. V., & Pereira, L. M. (2003). Semantic Web Logic Programming Tools. In *International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR'03)*, pp. 16–32.

Analyti, A., Antoniou, G., Damásio, C. V., & Wagner, G. (2004). Negation and Negative Information in the W3C Resource Description Framework. *Annals of Mathematics, Computing & Teleinformatics (AMCT)*, *1*(2), 25–34.

Analyti, A., Antoniou, G., Damásio, C. V., & Wagner, G. (2005). Stable Model Theory for Extended RDF Ontologies. In *4th International Semantic Web Conference (ISWC-2005)*, pp. 21–36.

Antoniou, G., Bikakis, A., & Wagner, G. (2004). A System for Nonmonotonic Rules on the Web. In *3rd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RULEML'03)*, pp. 23–36.

Antoniou, G., Billington, D., Governatori, G., & Maher, M. J. (2001). Representation Results for Defeasible Logic. *ACM Transactions on Computational Logic (TOCL)*, *2*(2), 255–287.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (Eds.). (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Bassiliades, N., Antoniou, G., & Vlahavas, I. P. (2004). DR-DEVICE: A Defeasible Logic System for the Semantic Web. In *2nd International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR'04)*, pp. 134–148.

Beckett, D. (2004). RDF/XML Syntax Specification (Revised). W3C Recommendation. Available at `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`.

Berger, R. (1966). The Undecidability of the Dominoe Problem. *Memoirs of the American Mathematical Society*, *66*, 1–72.

Berners-Lee, T. (1998). Design Issues - Architectual and Philosophical Points. Personal notes. Available at `http://www.w3.org/DesignIssues`.

Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., & Hendler, J. (2008). N3Logic: A Logical Framework For the World Wide Web. *to be published by Theory and Practice of Logic Programming (TPLP), Special Issue on Logic Programming and the Web.*

Bry, F., & Marchiori, M. (2005). Ten Theses on Logic Languages for the Semantic Web. In *3rd International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR-2005)*, pp. 42–49.

Damásio, C. V., Analyti, A., Antoniou, G., & Wagner, G. (2006). Supporting Open and Closed World Reasoning on the Web. In *4th Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR-2006)*, pp. 149–163.

de Bruijn, J., Franconi, E., & Tessaris, S. (2005). Logical Reconstruction of Normative RDF. In *OWL: Experiences and Directions Workshop (OWLED-2005)*, Galway, Ireland.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1998). $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, *10*(3), 227–252.

Donini, F. M., Nardi, D., & Rosati, R. (2002). Description Logics of Minimal Knowledge and Negation as Failure. *ACM Transactions on Computational Logic*, *3*(2), 177–225.

Eiter, T., Lukasiewicz, T., Schindlauer, R., & Tompits, H. (2004a). Combining Answer Set Programming with Description Logics for the Semantic Web. In *9th International Conference on Principles of Knowledge Representation and Reasoning (KR'04)*, pp. 141–151.

Eiter, T., Lukasiewicz, T., Schindlauer, R., & Tompits, H. (2004b). 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)*, pp. 81–97.

Eiter, T., Ianni, G., Polleres, A., & Schindlauer, R. (2006). Answer Set Programming for the Semantic Web. Tutorial co-located with the 3d European Semantic Web Conference (ESWC-2006).

Gelder, A. V., Ross, K. A., & Schlipf, J. S. (1991). The Well-Founded Semantics for General Logic Programs. *Journal of the ACM*, *38*(3), 620–650.

Gelfond, M., & Lifschitz, V. (1988). The Stable Model Semantics for Logic Programming. In Kowalski, R., & Bowen, K. A. (Eds.), *5th International Conference on Logic Programming*, pp. 1070–1080. MIT Press.

Gelfond, M., & Lifschitz, V. (1990). Logic programs with Classical Negation. In Warren, & Szeredi (Eds.), *7th International Conference on Logic Programming*, pp. 579–597. MIT Press.

Gelfond, M., & Lifschitz, V. (1991). Classical Negation in Logic programs and Disjunctive Databases. *New Generation Computing*, *9*, 365–385.

Hayes, P. (2004). RDF Semantics. W3C Recommendation. Available at `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`.

Herre, H., Jaspars, J., & Wagner, G. (1999). Partial Logics with Two Kinds of Negation as a Foundation of Knowledge-Based Reasoning. In Gabbay, D. M., & Wansing, H. (Eds.), *What Is Negation?* Kluwer Academic Publishers.

Herre, H., & Wagner, G. (1997). Stable Models are Generated by a Stable Chain. *Journal of Logic Programming*, *30*(2), 165–177.

Horrocks, I., & Patel-Schneider, P. F. (2003). Reducing OWL Entailment to Description Logic Satisfiability. In *2nd International Semantic Web Conference (ISWC-2003)*, pp. 17–29.

Horrocks, I., & Patel-Schneider, P. F. (2004). A Proposal for an OWL Rules Language. In *13th International Conference on World Wide Web (WWW'04)*, pp. 723–731. ACM Press.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission. Available at `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`.

Kifer, M., Lausen, G., & Wu, J. (1995). Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, *42*(4), 741–843.

Klyne, G., & Carroll, J. J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation. Available at `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

Levy, A. Y., & Rousset, M. (1998). Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, *104*(1-2), 165–209.

Lloyd, J. W., & Topor, R. W. (1984). Making Prolog more Expressive. *Journal of Logic Programming*, *1*(3), 225–240.

Maher, M. J. (2002). A Model-Theoretic Semantics for Defeasible Logic. In *ICLP 2002 Workshop on Paraconsistent Computational Logic (PCL-2002)*, pp. 255–287.

McGuinness, D. L., & van Harmelen, F. (2004). OWL Web Ontology Language Overview. W3C Recommendation. Available at `http://www.w3.org/TR/2004/REC-owl-features-20040210/`.

Motik, B., Sattler, U., & Studer, R. (2004). Query Answering for OWL-DL with Rules. In *3rd International Semantic Web Conference (ISWC-2004)*, pp. 549–563.

Patel-Schneider, P. F., Hayes, P., & Horrocks, I. (2004). OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation. Available at `http://www.w3.org/TR/2004/REC-owl-semantics-20040210/`.

Pereira, L. M., & Alferes, J. J. (1992). Well-Founded Semantics for Logic Programs with Explicit Negation. In Neumann, B. (Ed.), *European Conference on Artificial Intelligence*, pp. 102–106. John Wiley & Sons.

Prud'hommeaux, E., & Seaborne, A. (2008). SPARQL Query Language for RDF. W3C Recommendation. Available at `http://www.w3.org/TR/rdf-sparql-query/`.

Rao, P., Sagonas, K. F., Swift, T., Warren, D. S., & Freire, J. (1997). XSB: A System for Efficiently Computing WFS. In *Proceedings of 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'97)*, pp. 1070–1080.

Rosati, R. (1999). Towards Expressive KR Systems Integrating Datalog and Description Logics: Preliminary Report. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pp. 160–164.

Rosati, R. (2005). On the Decidability and Complexity of Integrating Ontologies and Rules. *Journal of Web Semantics*, *3*, 61–73.

Schaffert, S., Bry, F., Besnard, P., Decker, H., Decker, S., Enguix, C. F., & Herzig, A. (2005). Paraconsistent Reasoning for the Semantic Web. In *Workshop on Uncertainty Reasoning for the Semantic Web, co-located with ISWC-2005*, pp. 104–105.

Sintek, M., & Decker, S. (2002). TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *1st International Semantic Web Conference (ISWC-2002)*, pp. 364–378. Springer-Verlag.

ter Horst, H. J. (2004). Extending the RDFS Entailment Lemma. In *3rd International Semantic Web Conference (ISWC-2004)*, pp. 77–91.

ter Horst, H. J. (2005a). Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity. In *4th International Semantic Web Conference (ISWC-2005)*, pp. 668–684.

ter Horst, H. J. (2005b). Completeness, Decidability and Complexity of Entailment for RDF Schema and a Semantic Extension Involving the OWL Vocabulary. *Journal of Web Semantics*, *3*(2-3), 79–115.

Wagner, G. (1991). A Database Needs Two Kinds of Negation. In *3rd Symposium on Mathematical Fundamentals of Database and Knowledge Base Systems (MFDBS'91)*, pp. 357–371. Springer-Verlag.

Wagner, G. (2003). Web Rules Need Two Kinds of Negation. In *1st International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR'03)*, pp. 33–50. Springer-Verlag.

Wagner, G., Giurca, A., & Lukichev, S. (2005). A General Markup Framework for Integrity and Derivation Rules. In *Dagstuhl Seminar Proceedings: Principles and Practices of Semantic Web Reasoning*.

Wagner, G., Giurca, A., & Lukichev, S. (2006). A Usable Interchange Format for Rich Syntax Rules Integrating OCL, RuleML and SWRL. In *Workshop on Reasoning on the Web (RoW-2006), co-located with WWW-2006)*.

Yang, G., & Kifer, M. (2003a). 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)*, pp. 95–110.

Yang, G., & Kifer, M. (2003b). Reasoning about Anonymous Resources and Meta Statements on the Semantic Web. *Journal on Data Semantics*, *1*, 69–97.

Yang, G., Kifer, M., & Zhao, C. (2003). 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)*, pp. 671–688.