On the Computability and Complexity Issues of Extended RDF

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

[1] Institute of Computer Science, FORTH-ICS, Greece
[2] Department of Computer Science, University of Crete, Greece
[3] CENTRIA, Departamento de Informatica, Faculdade de Ciencias e Tecnologia, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
[4] Inst. of Informatics, Brandenburg Univ. of Technology at Cottbus, Germany
{analyti, antoniou}@ics.forth.gr, cd@di.fct.unl.pt, G.Wagner@tu-cottbus.de

Abstract. ERDF stable model semantics is a recently proposed semantics for ERDF ontologies and a faithful extension of RDFS semantics on RDF graphs. In this paper, we elaborate on the computability and complexity issues of the ERDF stable model semantics. We show that decidability under this semantics cannot be achieved, unless ERDF ontologies of restricted syntax are considered. Therefore, we propose a slightly modified semantics for ERDF ontologies, called *ERDF #n-stable model semantics*. We show that entailment under this semantics is in general decidable and it also extends RDFS entailment. An equivalence statement between the two semantics and various complexity results are provided.
Keywords: Extended RDF ontologies, Semantic Web, negation, rules, complexity.

1 Introduction

Rules constitute the next layer over the ontology languages of the Semantic Web, allowing arbitrary interaction of variables in the head and body of the rules. Berners-Lee [3] identifies the following fundamental theoretical problems: negation and contradictions, open-world versus closed-world assumptions, and rule systems for the Semantic Web. In [1], the Semantic Web language RDFS [8] is extended to accommodate the two negations of Partial Logic [9], namely *weak negation* $\sim$ (expressing negation-as-failure or non-truth) and *strong negation* $\neg$ (expressing explicit negative information or falsity), as well as derivation rules. The new language is called *Extended RDF* (*ERDF*). In [1], the *stable model semantics* of ERDF ontologies is developed, based on Partial Logic, extending the model-theoretic semantics of RDFS [8].

ERDF enables the combination of closed-world (non-monotonic) and open-world (monotonic) reasoning, in the same framework, through the presence of weak negation (in the body of the rules) and the new metaclasses *erdf*:*TotalClass* and *erdf*:*TotalProperty*, respectively. In particular, relating strong and weak

negation at the interpretation level, ERDF distinguishes two categories of properties and classes. *Partial properties* are properties p that may have truth-value gaps, that is $p(x, y)$ is possibly neither true nor false. *Total properties* are properties p that satisfy *totalness*, that is $p(x, y)$ is either true or false. Partial and total classes c are defined similarly, by replacing $p(x, y)$ by $rdf{:}type(x, c)$. ERDF also distinguishes between properties (and classes) that are completely represented in a knowledge base and those that are 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 $\neg rdf{:}type(?x, c) \leftarrow \sim rdf{:}type(?x, c)$. 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 $\neg p(x, y)$[5], 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.

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, with possibly all connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$, $\forall$, $\exists$ in the body of a rule, and strong negation $\neg$ in the head of a rule.

Example 1. We want to select wines for a dinner such that for each adult guest that (we know that) likes wine, there is on the table exactly one wine that he/she likes. Further, we want guests who are neither adults nor children to be served *Coca-Cola*. Additionally, we want adult guests, for whom we do not know if they like wine, also to be served *Coca-Cola*. Assume that in contrast to a child, we cannot decide if guest is an adult or not. For this drink selection problem, we use the classes: (i) $ex{:}Guest$, whose instances are the persons that will be invited to the dinner, (ii) $ex{:}Wine$, whose instances are wines, (iii) $ex{:}SelectedWine$ whose instances the wines *chosen* to be served, (iv) $ex{:}Adult$, whose instances are persons, 18 years of age or older, and (v) $ex{:}Child$, whose instances are persons, 10 years of age or younger. Additionally, we use the properties: (i) $ex{:}likes(X, Y)$ indicating that *we know that* person X likes wine Y, and (ii) $ex{:}serveSoftDrink(X, Y)$ indicating that person X will be served soft drink Y. An ERDF program P that describes this drink selection problem is the following[6,7]:

$$id(?x, ?x) \leftarrow true.$$

$$rdf{:}type(?y, SelectedWine) \leftarrow rdf{:}type(?x, Guest),\ rdf{:}type(?x, Adult),\ rdf{:}type(?y, Wine),\ likes(?x, ?y),$$

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

[6] To improve readability, we ignore the example namespace $ex{:}$.

[7] Commas "," in the body of the rules indicate conjunction $\wedge$.

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

$$rdf{:}type(Adult, erdf{:}TotalClass)\ \leftarrow\ true.$$

$$\neg rdf{:}type(?x, Child)\ \leftarrow\ {\sim}rdf{:}type(?x, Child).$$

$$serveSoftDrink(?x, Coca\text{-}Cola)\ \leftarrow\ rdf{:}type(?x, Guest), \neg rdf{:}type(?x, Adult), \neg rdf{:}type(?x, Child).$$

$$serveSoftDrink(?x, Coca\text{-}Cola)\ \leftarrow\ rdf{:}type(?x, Guest), rdf{:}type(?x, Adult), \forall ?y\ (rdf{:}type(?y, Wine)\ \supset\ {\sim}likes(?x, ?y)).$$

Consider now the ERDF graph G, containing the factual information: $G = \{rdf{:}type(Carlos, Guest)$, $rdf{:}type(Gerd, Guest)$, $rdf{:}type(Anne, Guest)$, $rdf{:}type(Riesling, Wine)$, $rdf{:}type(Retsina, Wine)$, $likes(Gerd, Riesling)$, $likes(Gerd, Retsina)$, $likes(Carlos, Retsina)$, $rdf{:}type(Gerd, Adult)$, $rdf{:}type(Carlos, Adult)\}$.

Then, $O = \langle G, P\rangle$ is an ERDF ontology. Note that *Adult* is declared in P as total class[8]. Thus, on this class the OWA applies and case-based reasoning on the truth value of $rdf{:}type(Anne, Adult)$ is performed. On the other hand, $likes(X, Y)$ is a partial property and *Child* is a partial class. In particular, on *Child* a CWA applies, expressed by a default closure rule. □

In [1], it is shown that stable model entailment conservatively extends RDFS entailment from RDF graphs to ERDF ontologies. Unfortunately, satisfiability and entailment under the ERDF stable model semantics are in general undecidable. In this work, we further elaborate on the undecidability result of the ERDF stable model semantics. We show that decidability cannot be achieved under this semantics, unless ERDF ontologies of restricted syntax are considered. This is due to the fact that the RDF vocabulary is infinite. Therefore, to achieve decidability of reasoning in the general case, we propose a modified semantics, called *ERDF #n-stable model semantics* (for $n \in \mathbb{N}$). The new semantics also extends RDFS entailment from RDF graphs to ERDF ontologies. Moreover, if O is a simple ERDF ontology (i.e., the bodies of the rules of O contain only the logical factors $\sim$, $\neg$, $\wedge$) then query answering under the ERDF #n-stable model semantics (for $n \in \mathbb{N}$) reduces to query answering under the answer set semantics [7]. An equivalence statement between the ERDF stable and #n-stable model semantics is provided. Moreover, we provide complexity results for (i) the ERDF #n-stable model semantics on simple ERDF ontologies and objective ERDF ontologies (i.e., ERDF ontologies whose rules contain only the logical factors $\neg$, $\wedge$) and (ii) the ERDF stable model semantics on objective ERDF ontologies.

The rest of the paper is organized as follows: Section 2 reviews the stable model semantics of ERDF ontologies. In Section 3, we propose the #n-stable model semantics of ERDF ontologies that extends RDFS entailment on RDF graphs and guarantees decidability of reasoning. Additionally, we provide an equivalence statement between the ERDF #n-stable and stable model semantics. Section 4 provides various complexity results for ERDF #n-stable and stable model semantics. Finally, Section 4 concludes the paper and reviews related work.

[8] Of course, this declaration could had been included (equivalently) in G, instead of P.

2 Stable Model Semantics of ERDF Ontologies

In this Section, we briefly review the stable model semantics of ERDF ontologies. Details and examples can be found in [1].

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 $\mathcal{LIT}$. We consider a set Var of variable symbols, such that the sets Var, $\mathcal{URI}$, $\mathcal{LIT}$ are pairwise disjoint. In our examples, variable symbols are prefixed by "?".

Let V be a vocabulary. An *ERDF triple* over V is an expression of the form $p(s, o)$ or $\neg p(s, o)$, where $s, o \in V \cup Var$ are called *subject* and *object*, respectively, and $p \in V \cap \mathcal{URI}$ is called *property*. 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.

Let V be a vocabulary. We denote by $L(V)$ the smallest set that contains the ERDF triples over V and is closed with respect to the following conditions: if $F, G \in L(V)$ then $\{\sim F,\ F \wedge G,\ F \vee G,\ F \supset G,\ \exists x F,\ \forall x F\} \subseteq L(V)$, where $x \in Var$. An *ERDF formula* over V is an element of $L(V)$. We denote the set of variables appearing in F by $Var(F)$, and the set of free variables appearing in F by $FVar(F)$. Moreover, we denote the set of URI references and literals appearing in F by V_F.

Intuitively, an ERDF graph G represents an existentially quantified conjunction of ERDF triples. Specifically, let $G = \{t_1, ..., t_m\}$ be an ERDF graph, and let $Var(G) = \{x_1, ..., x_k\}$. Then, G represents the ERDF formula $formula(G) = \exists ?x_1, ..., \exists ?x_k\ t_1 \wedge ... \wedge t_m$. Existentially quantified variables in ERDF graphs are handled by *skolemization*. 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 *skolemization* of G, denoted by $sk(G)$, is the ground ERDF graph derived from G after replacing each $x \in Var(G)$ by $sk_G(x)$.

An *ERDF rule* r over a vocabulary V is an expression of the form: $Concl(r) \leftarrow Cond(r)$, where $Cond(r) \in L(V) \cup \{true\}$ and $Concl(r)$ is an ERDF triple or *false*. We denote the set of variables and the set of free variables of r by $Var(r)$ and $FVar(r)$[9], respectively. An *ERDF program* P is a set of ERDF rules. We denote the set of URI references and literals appearing in P by V_P.

An *ERDF ontology* is a pair $O = \langle G, P \rangle$, where G is an ERDF graph and P is an ERDF program.

A partial interpretation is an extension of a simple interpretation of RDF semantics [8], where each property is associated not only with a truth extension but also with a falsity extension.

Definition 1 (Partial interpretation). A *partial interpretation* I of a vocabulary V consists of:

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

- A non-empty set of resources Res_I, a set of properties $Prop_I$, and a set of literal values $LV_I \subseteq Res_I$, which contains $V \cap \mathcal{PL}$.
- A vocabulary interpretation mapping: I_V: $V \cap \mathcal{URI} \rightarrow Res_I \cup Prop_I$.
- A property-truth extension mapping[10]: $PT_I : Prop_I \rightarrow \mathcal{P}(Res_I \times Res_I)$.
- A property-falsity extension mapping: $PF_I : Prop_I \rightarrow \mathcal{P}(Res_I \times Res_I)$.
- A mapping $IL_I : V \cap \mathcal{TL} \rightarrow Res_I$.

We define the mapping: $I : V \rightarrow Res_I \cup Prop_I$, called *denotation*, such that: (i) $I(x) = I_V(x),\ \forall x \in V \cap \mathcal{URI}$, (ii) $I(x) = x,\ \forall\ x \in V \cap \mathcal{PL}$, and (iii) $I(x) = IL_I(x)$, $\forall\ x \in V \cap \mathcal{TL}$. □

A partial interpretation I of a vocabulary V is *coherent* iff for all $x \in Prop_I$, $PT_I(x) \cap PF_I(x) = \emptyset$.

Let I be a partial interpretation of a vocabulary V and let v be a partial function $v : Var \rightarrow Res_I$ (called *valuation*). If $x \in Var$, we define $[I + v](x) = v(x)$. If $x \in V$, we define $[I + v](x) = I(x)$.

Definition 2. (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 $I, v \models F$ 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 $I, u \models G$. □

Let F be an ERDF formula, let G be an ERDF graph, and let I be a partial interpretation of a vocabulary V. We define: $I \models F$ iff for each mapping $v : Var(F) \rightarrow Res_I$, it holds that $I, v \models F$. Additionally, we define: $I \models G$ iff $I \models formula(G)$.

We assume that for every partial interpretation I, it holds that $I \models true$ and $I \not\models false$.

The vocabulary of RDF, $\mathcal{V}_{RDF}$, is a set of $\mathcal{URI}$ references in the *rdf*: namespace [8]. The vocabulary of RDFS, $\mathcal{V}_{RDFS}$, is a set of $\mathcal{URI}$ references in the *rdfs*: namespace [8]. The *vocabulary of ERDF* is defined as $\mathcal{V}_{ERDF} = \{$*erdf*:*TotalClass*, *erdf*:*TotalProperty*$\}$. Intuitively, instances of the metaclass *erdf*:*TotalClass* are classes c that satisfy totalness, meaning that each resource x belongs either to the truth or falsity extension of c (i.e., the statement "x is of type c" is either

[10] The notation $\mathcal{P}(S)$, where S is a set, denotes the *powerset* of S.

true or explicitly false). Similarly, instances of the metaclass *erdf*:*TotalProperty* are properties p that satisfy totalness, meaning that each pair of resources $\langle x, y\rangle$ belongs either to the truth or falsity extension of p (i.e., the statement "$\langle x, y\rangle$ satisfies property p" is either true or explicitly false).

Definition 3 (ERDF interpretation). An *ERDF interpretation* I of a vocabulary V is a coherent, partial interpretation of $V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$, extended by the new ontological categories $Cls_I \subseteq Res_I$ for classes, $TCls_I \subseteq Cls_I$ for total classes, and $TProp_I \subseteq Prop_I$ for total properties, as well as the class-truth extension mapping $CT_I : Cls_I \to \mathcal{P}(Res_I)$, and the class-falsity extension mapping $CF_I : Cls_I \to \mathcal{P}(Res_I)$, such that:

1. $x \in CT_I(y)$ iff $\langle x, y\rangle \in PT_I(I(rdf{:}type))$, and
   $x \in CF_I(y)$ iff $\langle x, y\rangle \in PF_I(I(rdf{:}type))$.
2. The ontological categories are defined as follows:
   $Prop_I = CT_I(I(rdf{:}Property))$ $\quad Cls_I = CT_I(I(rdfs{:}Class))$
   $Res_I = CT_I(I(rdfs{:}Resource))$ $\quad LV_I = CT_I(I(rdfs{:}Literal))$
   $TCls_I = CT_I(I(erdf{:}TotalClass))$ $\quad TProp_I = CT_I(I(erdf{:}TotalProperty))$.
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 [8], respectively.
17. I satisfies the following triples, called *ERDF axiomatic triples*:
    $rdfs{:}subClassOf(erdf{:}TotalClass, rdfs{:}Class)$.
    $rdfs{:}subClassOf(erdf{:}TotalProperty, rdfs{:}Class)$. □

The *vocabulary* of an ERDF ontology O is defined as $V_O = V_{sk(G)} \cup V_P \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Additionally, we denote by Res_O^H 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.

Definition 4 (Herbrand interpretation 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: (i) $Res_I = Res_O^H$, (ii) $I_V(x) = x$, for all $x \in V_O \cap \mathcal{URI}$, (iii) $IL_I(x) = x$, 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 $\mathcal{I}^H(O)$. □

Let $O = \langle G, P\rangle$ be an ERDF ontology and let $I, J \in \mathcal{I}^H(O)$. We say that J *extends* I, denoted by $I \leq J$, iff $Prop_I \subseteq Prop_J$, and $\forall\ p \in Prop_I,\ \ PT_I(p) \subseteq PT_J(p)$ and $PF_I(p) \subseteq PF_J(p)$.

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 $[P]_V = \bigcup_{r\in P}[r]_V$.

Below, we define the stable models of an ERDF ontology, based on the coherent stable models of Partial Logic [9].

Definition 5 (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:

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 $\mathcal{M}^{st}(O)$. □

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 stable model M.

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

Example 2. Consider the ERDF ontology O of Example 1. Then, O has two stable models M_1, M_2, where $M_1 \models \neg$*rdf:type*(*Anne*, *Adult*) and $M_2 \models$ *rdf:type*(*Anne*, *Adult*)[11]. For both $M \in \{M_1, M_2\}$, it holds $M \models$ *serveSoftDrink*(*Anne*, *Coca-Cola*). This is because, if *Anne* is not an adult then, since she is not a child, it is decided to drink *Coca-Cola*. If *Anne* is an adult then, since it is not known if she likes wine, it is also decided to drink *Coca-Cola*. Thus, it holds $O \models^{st}$ *serveSoftDrink*(*Anne*, *Coca-Cola*). Additionally, for both $M \in \{M_1, M_2\}$, it holds $M \models$ *rdf:type*(*Retsina*, *SelectedWine*) $\wedge$ $\sim$*rdf:type*(*Riesling*, *SelectedWine*). This is because (i) both *Gerd* and *Carlos* like *Retsina* and (ii)

[11] Note that *ex:Adult* is a total class and that we do not know if Anne is an adult.

Carlos likes *only Retsina*. Thus, it holds $O \models^{st} rdf{:}type(Retsina, SelectedWine) \wedge \sim rdf{:}type(Riesling, SelectedWine)$. □

In [1], it is shown that stable model entailment conservatively extends RDFS entailment from RDF graphs to ERDF ontologies.

Proposition 1. 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 $\langle G, \emptyset \rangle \models^{st} G'$.

3 Undecidability of ERDF Stable Model Semantics leads to #*n*-Stable Model Semantics

Unfortunately, satisfiability and entailment under the ERDF stable model semantics are in general undecidable [1]. The proof of undecidability exploits a reduction from the *unbounded tiling problem*, whose existence of a solution is known to be undecidable [2]. 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.

Definition 6 (Simple, Objective ERDF ontology). An ERDF formula F is called *simple* if it has the form $t_1 \wedge ... \wedge t_k \wedge \sim t_{k+1} \wedge ... \wedge \sim t_m$, where each t_i, $i = 1, ..., m$, is an ERDF triple. An ERDF program P is called *simple* if for all $r \in P$, $Cond(r)$ is a simple ERDF formula or *true*. An ERDF ontology $O = \langle G, P \rangle$ is called *simple*, if P is a simple ERDF program. A simple ERDF ontology O (resp. ERDF program P) is called *objective*, if no weak negation appears in O (resp. P). □

Reduction in [1] shows that ERDF stable model satisfiability and entailment remain undecidable, even if (i) $O = \langle G, P \rangle$ is a simple ERDF ontology, (ii) the terms $erdf{:}TotalClass$ and $erdf{:}TotalProperty$ do not appear in O (i.e., $(V_G \cup V_P) \cap \mathcal{V}_{ERDF} = \emptyset$), and (iii) the entailed formula has the form $\exists \bar{x}\ F$, where F is a simple ERDF formula and $\bar{x}$ are the variables appearing in F. Moreover, we can prove by a reduction from the unbounded tiling problem [2] that even if $O = \langle G, P \rangle$ is an objective ERDF ontology, entailment of a *general* ERDF formula F under the ERDF stable model semantics is still undecidable.

Let O be a *general* ERDF ontology. The source of undecidability of the ERDF stable model semantics of O is the fact that $\mathcal{V}_{RDF}$ is infinite. Thus, the vocabulary of O is also infinite (note that $\{rdf{:}\_i \mid i \geq 1\} \subseteq \mathcal{V}_{RDF} \subseteq V_O$). In this Section, we slightly modify the definition of the ERDF stable model semantics, based on a redefinition of the vocabulary of an ERDF ontology, which now becomes finite. We call the modified semantics, the *ERDF #n-stable model semantics* (for $n \in \mathbb{N}$).

In order to define the ERDF #n-stable model semantics, we need to modify several of the definitions on which the ERDF stable model semantics is based. Specifically:

- We define: $\mathcal{V}_{RDF}^{\#n} = \mathcal{V}_{RDF} - \{rdf{:}\_i \mid i > n\}$.
- An *ERDF #n-interpretation* is defined exactly as an ERDF interpretation in Def. 3 except that $\mathcal{V}_{RDF}$ is replaced by $\mathcal{V}_{RDF}^{\#n}$ and in semantic condition 16, only the RDF and RDFS axiomatic triples that contain $\mathcal{URI}$ references in $\mathcal{V}_{RDF}^{\#n}$ are considered.
- Let $O = \langle G, P\rangle$ be an ERDF ontology. We define: $V_O^{\#n} = V_O - \{rdf{:}\_i \mid i > n\}$, and $Res_O^{H_{\#n}} = Res_O^H - \{rdf{:}\_i \mid i > n\}$.
- Let $O = \langle G, P\rangle$ be an ERDF ontology. An *#n-Herbrand interpretation* I of O is an ERDF #n-interpretation of $V_O^{\#n}$ such that: (i) $Res_I = Res_O^{H_{\#n}}$, (ii) $I_V(x) = x$, for all $x \in V_O^{\#n} \cap \mathcal{URI}$, (iii) $IL_I(x) = x$, if x is a typed literal in $V_O^{\#n}$ 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 $V_O^{\#n}$. We denote the set of #n-Herbrand interpretations of O by $\mathcal{I}^{H_{\#n}}(O)$.
- Let $O = \langle G, P\rangle$ be an ERDF ontology. An *(ERDF) #n-stable model* of O is defined as a stable model of O in Def. 5, except that $\mathcal{I}^H(O)$ is replaced by $\mathcal{I}^{H_{\#n}}(O)$ and V_O is replaced by $V_O^{\#n}$. The set of #n-stable models of O is denoted by $\mathcal{M}^{st_{\#n}}(O)$.

Let $O = \langle G, P\rangle$ be an ERDF ontology and let F be an ERDF formula or ERDF graph. Let $n \in \mathbb{N}$. We say that O *entails* F under the *(ERDF) #n-stable model semantics*, denoted by $O \models^{st_{\#n}} F$ iff for all $M \in \mathcal{M}^{st_{\#n}}(O)$, $M \models F$.

Let $O = \langle G, P\rangle$ be an ERDF ontology and let F be an ERDF formula. Let $n \in \mathbb{N}$. The *(ERDF) #n-stable answers* of F w.r.t. O are defined as follows[12]:

$$Ans_O^{st_{\#n}}(F) = \begin{cases} \text{“yes”} & \text{if } FVar(F) = \emptyset \text{ and } \forall M \in \mathcal{M}^{st_{\#n}}(O) : M \models F \\ \text{“no”} & \text{if } FVar(F) = \emptyset \text{ and } \exists M \in \mathcal{M}^{st_{\#n}}(O) : M \not\models F \\ \{v : FVar(F) \rightarrow V_O^{\#n} \mid \forall M \in \mathcal{M}^{st_{\#n}}(O) : M \models v(F)\} & \\ \qquad \text{if } FVar(F) \neq \emptyset & \end{cases}$$

Let $O = \langle G, P\rangle$ be an ERDF ontology. We define: (i) $n_O = 0$, if $(V_G \cup V_P) \cap \{rdf{:}\_i \mid i \geq 1\} = \emptyset$, and (ii) $n_O = max(\{i \in \mathbb{N} \mid rdf{:}\_i \in V_G \cup V_P\})$, otherwise.

For example, if O is the ERDF ontology of Example 1 then $n_O = 0$.

Proposition 2 below relates stable model entailment and #n-stable model entailment. First, we provide a definition. 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) $FVar(F) = \emptyset$. For example, let $F = (\exists ?x\ rdf{:}type(?x, Vertex) \wedge rdf{:}type(?x, Red)) \vee (\exists ?x\ rdf{:}type(?x, Vertex) \wedge \neg rdf{:}type(?x, Blue))$. 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 2. Let $O = \langle G, P\rangle$ be an objective ERDF ontology and let $n \geq max(n_O, 1)$. Let F^d be an ERDF *d*-formula s.t. $max(\{i \in \mathbb{N} \mid rdf{:}\_i \in V_{F_d}\}) \leq n$. It holds: $O \models^{st} F^d$ iff $O \models^{st_{\#n}} F^d$.

[12] $v(F)$ results from F after replacing all the free variables x in F by $v(x)$.

Since $V_O^{\#n}$ (for $n \in \mathbb{N}$) is finite, query answering under the ERDF $\#n$-stable model semantics is decidable. Now, since satisfiability under the ERDF stable model semantics is in general undecidable, Proposition 2 does not hold in the case that $O = \langle G, P \rangle$ is a general ERDF ontology. Moreover, Proposition 2 does not hold in the case that F is a general ERDF formula. For example, consider the ERDF graph G:

$$G = \{rdf{:}type(x, c1) \mid x \in \{c1, c2, id\} \cup \mathcal{V}_{RDF}^{\#0} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}\}$$

Additionally, consider the ERDF program $P = \{id(?x, ?x) \leftarrow true.\}$ and the ERDF formula F (which is not an ERDF d-formula):

$F = \exists ?x, \exists ?y \sim rdf{:}type(?x, c1) \wedge \sim rdf{:}type(?y, c1) \wedge \sim id(?x, ?y).$

Let $O = \langle G, P \rangle$. It holds, $n_O = 0$. Note that $O \models^{st} F$, while $O \not\models^{st_{\#1}} F$.

The following proposition is a direct consequence of Propositions 1 and 2, and shows that $\#n$-stable model entailment also extends RDFS entailment from RDF graphs to ERDF ontologies.

Proposition 3. Let G, G' be RDF graphs such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$, $V_{G'} \cap \mathcal{V}_{ERDF} = \emptyset$, and $V_{G'} \cap sk_G(Var(G)) = \emptyset$. Let $O = \langle G, \emptyset \rangle$ and $n \geq max(n_O, 1)$. If $max(\{i \in \mathbb{N} \mid rdf{:}\_i \in V_{G'}\}) \leq n$ then: $G \models^{RDFS} G'$ iff $O \models^{st_{\#n}} G'$.

4 Complexity Results

In this section, we provide complexity results for (i) the ERDF $\#n$-stable model semantics on simple and objective ERDF ontologies, and (ii) the ERDF stable model semantics on objective ERDF ontologies. Additionally, for $n \in \mathbb{N}$, we show that the $\#n$-stable answers of a simple ERDF formula F w.r.t. a simple ERDF ontology $O = \langle G, P \rangle$ can be computed through Answer Set Programming [7] on an extended logic program (ELP) $\Pi_O^{\#n}$ (not given here due to space limitations).

Let Π be an extended logic program (ELP) and let F be a query of the form $L_1 \wedge ... \wedge L_k \wedge \sim L_{k+1} \wedge ... \wedge \sim L_m$, where $L_i, i = 1, ..., m$, is an ELP literal. We will denote by $Ans_\Pi^{AS}(F)$ the (skeptical) answers of F w.r.t. Π according to answer set semantics [7].

Proposition 4. Let $O = \langle G, P \rangle$ be a simple ERDF ontology and let F be a simple ERDF formula. Let $n \in \mathbb{N}$. It exists an ELP $\Pi_O^{\#n}$ generated in polynomial time w.r.t. the size of O and n s.t. $Ans_O^{st_{\#n}}(F) = Ans_{\Pi_O^{\#n}}^{AS}(F')$, where F' is the query that results after replacing each ERDF triple $p(s, o)$ appearing in F by the ELP literal $Holds(s, p, o)$.

Based on Proposition 4 and complexity results for answer set semantics (see [5]), we can state the following Corollary.

Corollary 1. Let $O = \langle G, P \rangle$ be a simple ERDF ontology and let F be an ERDF formula. Additionally, let v be (i) one of {"yes", "no"}, if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \in \mathbb{N}$.

1. The problem of establishing whether O has an $\#n$-stable model is NP-complete w.r.t. size of $sk(G) \cup [P]_{V_O^{\#n}}$.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is co-NP-complete w.r.t. size of $sk(G) \cup [P]_{V_O^{\#n}}$.

Below, we state complexity results for the $\#n$-stable model semantics of objective ERDF ontologies. We see that even though no weak negation appears in the rules of objective ERDF ontologies, complexity of reasoning w.r.t. simple ERDF ontologies remains the same. This is due to the ERDF metaclasses *erdf*:*TotalClass* and *erdf*:*TotalProperty* on the instances of which, the OWA applies.

Proposition 5. Let $O = \langle G, P \rangle$ be an objective ERDF ontology. Let G' be an ERDF graph and let F be an ERDF formula. Additionally, let v be (i) one of {"yes", "no"}, if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \in \mathbb{N}$.

1. The problem of establishing whether O has an $\#n$-stable model is NP-complete w.r.t. size of $sk(G) \cup [P]_{V_O^{\#n}}$.
2. The problems of establishing whether: (i) $O \models^{st_{\#n}} G'$ and (ii) $O \models^{st_{\#n}} F$ are co-NP-complete w.r.t. size of $sk(G) \cup [P]_{V_O^{\#n}}$.

The hardness part of the above complexity results can be proved by a reduction from the *Graph 3-Colorability* problem, which is a classical NP-complete problem.

Based on Proposition 2 and Proposition 5, it follows:

Corollary 2. Let $O = \langle G, P \rangle$ be an objective ERDF ontology. Let G' be an ERDF graph and let F^d be an ERDF d-formula s.t. $max(\{i \in \mathbb{N} \mid rdf{:}\_i \in V_{\mathtt{X}}\}) \leq n_O$, where $\mathtt{X} \in \{G', F_d\}$.

1. The problem of establishing whether O has a stable model is NP-complete w.r.t. size of $sk(G) \cup [P]_{V_O^{\#n_O}}$.
2. The problems of establishing whether: (i) $O \models^{st} G'$ and (ii) $O \models^{st} F^d$ are co-NP-complete w.r.t. size of $sk(G) \cup [P]_{V_O^{\#n_O}}$.

Yet, as mentioned in Section 3, satisfiability and entailment of *simple* (and of course, general) ERDF ontologies under the ERDF stable model semantics are undecidable.

5 Conclusions & Related Work

In this paper, we elaborated on the computability and complexity issues of the stable model semantics of ERDF ontologies. We show that decidability under this semantics cannot be achieved, unless ERDF ontologies of restricted syntax are considered. We propose the *#n-stable model semantics* of ERDF ontologies

(for $n \in \mathbb{N}$) and show that entailment under this semantics extends RDFS entailment. Moreover, query answering under the ERDF $\#n$-stable model semantics is decidable. An equivalence statement between the ERDF stable and $\#n$-stable model semantics, as well as various complexity results are provided. Future work concerns the implementation of the $\#n$-stable model semantics on ERDF ontologies, as well as the extension of our complexity results to other syntax restricted ERDF ontologies and general ERDF ontologies.

Notation 3 (N3) [4] provides a more human readable syntax for RDF and also extends RDF by adding numerous pre-defined constructs for being able to express rules conveniently. In particular, N3 contains a built-in (`log:notIncludes`) for expressing simple negation-as-failure tests and another built-in (`log:definitiveDocument`) for making restricted completeness assumptions. However, N3 does not provide strong negation and closed-world reasoning is not fully supported. In [11], RDF graphs are extended with a set of rules R and R-entailment is defined, extending RDFS entailment. However, in this work, weak and strong negation are not considered. In [6], RDFS is extended with rules and/or general axioms, using embeddings in F-Logic [10]. However, such extensions are not entirely faithful to the model-theoretic semantics of RDF.

References

1. A. Analyti, G. Antoniou, C. V. Damásio, and G. Wagner. Extended RDF as a Semantic Foundation of Rule Markup Languages. *Journal of Artificial Intelligence Research (JAIR)*, 32:37–94, 2008.
2. R. Berger. The Undecidability of the Dominoe Problem. *Memoirs of the American Mathematical Society*, 66:1–72, 1966.
3. T. Berners-Lee. Design Issues - Architectual and Philosophical Points. Personal notes, 1998. Available at `http://www.w3.org/DesignIssues`.
4. T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, and J. Hendler. N3Logic: A Logical Framework For the World Wide Web. *Theory and Practice of Logic Programming (TPLP)*, 8(3):249–269, 2008.
5. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.
6. J. de Bruijn and S. Heymans. RDF and Logic: Reasoning and Extension. In *6th International Workshop on Web Semantics (WebS'07), co-located with DEXA-2007*, pages 460–464, 2007.
7. M. Gelfond and V. Lifschitz. Logic programs with Classical Negation. In *7th International Conference on Logic Programming*, pages 579–597, 1990.
8. P. Hayes. RDF Semantics. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`.
9. H. Herre, J. Jaspars, and G. Wagner. Partial Logics with Two Kinds of Negation as a Foundation of Knowledge-Based Reasoning. In D. M. Gabbay and H. Wansing, editors, *What Is Negation?* Kluwer Academic Publishers, 1999.
10. M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, 1995.
11. H. J. ter Horst. Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity. In *4th International Semantic Web Conference (ISWC-2005)*, pages 668–684, November 2005.