Extended RDF: Computability and Complexity Issues

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

[1] Institute of Computer Science, FORTH-ICS, Greece
[2] CENTRIA, Departamento de Informatica, Faculdade de Ciencias e Tecnologia, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
[3] Department of Informatics, University of Huddersfield, UK
analyti@ics.forth.gr,cd@fct.unl.pt, G.Antoniou@hud.ac.uk

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. Based on the undecidability result of ERDF stable model semantics, 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 also extends RDFS entailment. Equivalence statements between the two semantics are provided. Additionally, we provide algorithms that compute the ERDF #n-stable models of syntax-restricted and general ERDF ontologies. Further, we provide complexity results for the ERDF #n-stable model semantics on syntax-restricted and general ERDF ontologies. Finally, we provide complexity results for the ERDF stable model semantics on syntax-restricted ERDF ontologies.
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. In particular, Berners-Lee [12] identifies the following fundamental theoretical problems: negation and contradictions, open-world versus closed-world assumptions, and rule systems for the Semantic Web. Therefore, in [6], the Semantic Web language RDFS [38, 32] is extended to accommodate the two negations of Partial Logic [33], 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. In particular, users can now add negative triples to RDF graphs. The new language is called *Extended RDF* (*ERDF*). In [6], the *ERDF stable model semantics* of ERDF ontologies is developed, based on Partial Logic, extending the model-theoretic semantics of RDFS [32].

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*:*TotalProperty* and *erdf*:*TotalClass*, 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 of the considered interpretations will satisfy $\neg p(x, y)$[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. We would like to note that in ERDF properties and classes are partial by default.

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. Examples of ERDF ontologies are provided in the next section.

In [6], it is shown that ERDF stable model entailment conservatively extends RDFS entailment from RDF graphs to ERDF ontologies. Unfortunately, as shown in [6], satisfiability and entailment under the ERDF stable model semantics are in general undecidable. In this work:

- In Section 4, we further elaborate on the undecidability result of the ERDF stable model semantics. Decidability cannot be achieved under this semantics, unless ERDF ontologies of restricted syntax are considered. Undecidability which is obtained even with the single presence of weak and strong negation in an ERDF ontology is due to the fact that the RDF vocabulary is infinite. Therefore, we propose in Section 5 a modified semantics, called *ERDF #n-stable model semantics* (for $n \in \mathbb{N}$), that achieves decidability of reasoning in the general case. The new semantics also extends RDFS entailment from RDF graphs to ERDF ontologies. Equivalence statements between the (original) ERDF stable and #n-stable model semantics for objective ERDF ontologies (i.e., ERDF ontologies whose rules contain only the logical factors $\neg$, $\wedge$) are provided.
- In Section 6, we show that 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 reduces to query answering under the answer set semantics [30] over a particular transformed program generated from O.
- In Section 6 and Section 8, we provide algorithms that compute the ERDF #n-stable models of simple and general ERDF ontologies, respectively.

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

– Using the previous algorithms, in Section 7 and Section 9, we provide model existence and query answering complexity results for the ERDF $\#n$-stable model semantics on objective and simple ERDF ontologies, ERDF ontologies without quantifiers, and general ERDF ontologies. We consider both the case that an ERDF ontology is bounded (i.e., there are no free variables in the body and head of its rules) or is not bounded. In particular, the complexity of query answering under the ERDF $\#n$-stable model semantics (i) on bounded objective and bounded simple ERDF ontologies is co-NP-complete, (ii) on objective and simple ERDF ontologies, as well as on bounded ERDF ontologies without quantifiers and ERDF ontologies without quantifiers, is Π_2^P=co-NP$^{\text{NP}}$-complete, and (iii) on bounded ERDF ontologies and general ERDF ontologies is PSPACE-complete. All previous results are with respect to the size of the ERDF ontology. Additionally, in Section 10, we provide combined complexity results (i.e., with respect to both the size of the ERDF ontology and the size of the query formula) of query answering. In particular, in the case that the query formula does not have quantifiers, the combined complexity of query answering is the same as the complexity of query answering with respect to the size of ERDF ontology, for all kinds of ERDF ontologies. Further, in the case that the query formula is general, the combined complexity of query answering is PSPACE-complete, for all kinds of ERDF ontologies. These complexity results are summarized in Table 1. We see that when the complexity of the structure of an ERDF ontology increases, its query answering complexity also increases though there are overlaps. The combined complexity of query answering for a general query formula F is the same for all kinds of ontologies due to the complexity of F.
– Finally, in Section 7, we provide complexity results for the (original) ERDF stable model semantics when restricted to bounded objective and objective ERDF ontologies. In particular, the complexity of query answering under the ERDF stable model semantics on bounded objective ERDF ontologies is co-NP-complete and on objective ERDF ontologies is Π_2^P-complete, provided that the query formula is an ERDF d-formula (i.e. a disjunction of existentially quantified conjunctions of ERDF triples, without free variables). This is due to the ERDF metaclasses $erdf{:}TotalClass$ and $erdf{:}TotalProperty$, on the instances of which the OWA applies.

This work extends our conference paper [7] (i) by showing that the ERDF stable model semantics is undecidable even on objective ERDF ontologies, (ii) by showing that if O is a simple ERDF ontology then query answering under the ERDF $\#n$-stable model semantics reduces to query answering under the answer set semantics, (iii) by providing algorithms that compute the ERDF $\#n$-stable models of simple and general ERDF ontologies, (iv) by providing complexity results for the ERDF $\#n$-stable model semantics on bounded simple ERDF ontologies, bounded objective ERDF ontologies, bounded ERDF ontologies without quantifiers, ERDF ontologies without quantifiers, bounded ERDF ontologies, and general ERDF ontologies, and (v) by extending related work and providing proofs for all propositions.

The rest of the paper is organized as follows: In Section 2, we present two examples of ERDF ontologies. Section 3 reviews the ERDF stable model semantics of ERDF ontologies. In Section 4, we show that the ERDF stable model semantics is undecidable even on objective ERDF ontologies. In Section 5, we propose the ERDF $\#n$-stable model semantics of ERDF ontologies that extends RDFS entailment on RDF graphs and guarantees decidability of reasoning. Additionally, we provide equivalence

ERDF ontology	model existence w.r.t. the size of the ERDF ontology	query answering w.r.t. the size of the ERDF ontology	combined complexity of query answering for an unquantified query formula	combined complexity of query answering for a general query formula
bounded objective	NP (Prop. 9.1)	co-NP (Prop. 9.3)	co-NP (Prop. 19.1)	PSPACE (Prop. 19.2)
objective	NP^{NP} (Prop. 12.1)	co-NP^{NP} (Prop. 12.2)	co-NP^{NP} (Prop. 19.1)	PSPACE (Prop. 19.2)
bounded simple	NP (Prop. 7)	co-NP (Prop. 8)	co-NP (Prop. 19.1)	PSPACE (Prop. 19.2)
simple	NP^{NP} (Prop. 11.1)	co-NP^{NP} (Prop. 11.2)	co-NP^{NP} (Prop. 19.1)	PSPACE (Prop. 19.2)
bounded without quantifiers	NP^{NP} (Prop.14.1)	co-NP^{NP} (Prop.14.2)	co-NP^{NP} (Prop. 19.1)	PSPACE (Prop. 19.2)
without quantifiers	NP^{NP} (Prop.15.1)	co-NP^{NP} (Prop.15.2)	co-NP^{NP} (Prop. 19.1)	PSPACE (Prop. 19.2)
bounded	PSPACE (Prop. 17.1)	PSPACE (Prop. 17.2)	PSPACE (Prop. 19.1)	PSPACE (Prop. 19.2)
general	PSPACE (Prop. 18.1)	PSPACE (Prop. 18.2)	PSPACE (Prop. 19.1)	PSPACE (Prop. 19.2)

Table 1. Complexity of ERDF #n-stable model semantics (all entries are completeness results)

statements between the ERDF #n-stable and the (original) ERDF stable model semantics. Section 6 considers query answering on simple ERDF ontologies under the ERDF #n-stable model semantics. Section 7 provides complexity results w.r.t. the size of the ERDF ontology for (i) the ERDF #n-stable model semantics on simple and objective ERDF ontologies, and (ii) the (original) ERDF stable model semantics on objective ERDF ontologies. Section 8 provides an algorithm that computes the ERDF #n-stable models of general ERDF ontologies. Section 9 provides complexity results w.r.t. the size of the ERDF ontology for the ERDF #n-stable model semantics on ERDF ontologies without quantifiers and general ERDF ontologies. Section 10 provides combined complexity results for all kinds of ERDF ontologies. Section 11 reviews related work. Finally, Section 12 concludes the paper. All proofs are provided in the Appendix A. Appendix B contains a list of symbols. Appendix C overviews RDFS reasoning.

2 Examples of ERDF ontologies

In this Section, we present two examples of ERDF ontologies.

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 a 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 are 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[5,6]:

$$(1)\ id(?x, ?x) \leftarrow true.$$
$$(2)\ rdf{:}type(?y, SelectedWine) \leftarrow rdf{:}type(?x, Guest),\ rdf{:}type(?x, Adult),\ rdf{:}type(?y, Wine), likes(?x, ?y),\ \forall ?z\ (rdf{:}type(?z, SelectedWine), {\sim}id(?y, ?z) \supset {\sim}likes(?x, ?z)).$$

$$(3)\ rdf{:}type(Adult, erdf{:}TotalClass) \leftarrow true.$$
$$(4)\ \neg rdf{:}type(?x, Child) \leftarrow {\sim}rdf{:}type(?x, Child).$$

$$(5)\ serveSoftDrink(?x, Coca\text{-}Cola) \leftarrow rdf{:}type(?x, Guest), \neg rdf{:}type(?x, Adult), \neg rdf{:}type(?x, Child).$$
$$(6)\ serveSoftDrink(?x, Coca\text{-}Cola) \leftarrow rdf{:}type(?x, Guest), rdf{:}type(?x, Adult), \forall ?y\ (rdf{:}type(?y, Wine) \supset {\sim}likes(?x, ?y)).$$

[5] To improve readability, we ignore the example namespace ex:.
[6] Commas "," in the body of the rules indicate conjunction $\wedge$.

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

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

Then, $O = \langle G, P\rangle$ is an ERDF ontology. Note that *Adult* is declared in P as a total class[7]. Thus, on this class the OWA applies and case-based reasoning on the truth value of $rdf{:}type(Anne, Adult)$ is performed. This is because somebody is either an *Adult* or is not and O does not contain complete knowledge about *Adult*. On the other hand, $likes(X, Y)$ is a partial property because somebody may neither like a wine and neither dislike it. *Child* is not a partial class because somebody is either a *Child* or is not. However, O contains complete knowledge about *Child*. Thus, on *Child* a CWA applies, expressed by a default closure rule in P (in line (4)). □

Example 2. Assume that a new drug for the Parkinson disease has been invented by a pharmaceutical company with the name *Steron*. Before it is released into the market it should pass preliminary tests on Parkinson patients. Assume that there exists a team of Parkinson patients that take the drug and all of them should be able to move each part of their body and not present dizziness or instability in order for the new drug to be effective. This is expressed by the following ERDF ontology $O = \langle G, P\rangle$, where $P =$

(1) $rdf{:}type(Steron, EffectiveParkisonDrug) \leftarrow \forall ?x\ belongs(?x, SteronTestTeam) \supset$
$\sim(\exists ?y\ rdf{:}type(?y, bodyPart), \neg move(?x, ?y)), \sim presents(?x, instability),$
$\sim presents(?x, dizziness).$

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

$G = \{rdf{:}type(arm, BodyPart), rdf{:}type(leg, BodyPart),$
$belongs(Anne, SteronTestTeam), belongs(Mary, SteronTestTeam),$
$belongs(Peter, SteronTestTeam), belongs(Joan, SteronTestTeam),$
$\neg move(Anne, arm), presents(Joan, instability)\}.$

Since *Anne* and *Joan* belong to the test team for *Steron* and *Anne* cannot move her arm and *Joan* presents instability, *Steron* is not considered an effective drug. □

3 Stable Model Semantics of ERDF Ontologies

In this Section, we provide the basic definitions of the ERDF stable model semantics of ERDF ontologies. More details and examples can be found in [6].

According to RDF concepts [38, 32], *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.

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

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 (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 $\{\sim F,\ \neg F,\ F \wedge G,\ F \vee G,\ F \supset G,\ \exists x F,\ \forall x F\} \subseteq L(V)$, where $x \in Var$. An *ERDF formula* over V is an element of $L(V)$. We denote the set of variables appearing in the ERDF formula 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*, where existential variables (blank nodes) are replaced by skolem constants. We will denote the skolemized graph G, by $sk(G)$.

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*. Without loss of generality, we assume that no bound variable in $Cond(r)$ appears free in $Concl(r)$. We denote the set of variables and the set of free variables of r by $Var(r)$ and $FVar(r)$[8], respectively. An *ERDF program* P is a finite 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.

We now define three kinds of ERDF ontologies that are going to be used in the paper.

Definition 1 (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 a (positive or negative) 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). □

Definition 2 (Bounded ERDF ontology). Let $O = \langle G, P \rangle$ be an ERDF ontology. We say that O is *bounded* if for each $r \in P$, there are no free variables in $Cond(r)$ and $Concl(r)$, i.e. $FVar(Cond(r)) = \emptyset$ and $FVar(Concl(r)) = \emptyset$. □

[8] $FVar(r) = FVar(Concl(r)) \cup FVar(Cond(r))$.

Notice that the notion of bounded ERDF ontology is a generalization of grounded logic programming rules, i.e. with no occurrence of variables, to general quantified ERDF formulas in rule conditions. In fact, bounded simple and objective ontologies can be equivalently defined as grounded simple and objective ontologies, respectively.

In the subsequent sections, we are going to define query answering according to $\#n$-stable model semantics and providing complexity of query answering for all kinds on ERDF ontologies both with respect to the size of the ERDF ontology and combined complexity (that is both w.r.t. the size of the ERDF ontology and the size of the query ERDF formula).

A partial interpretation is an extension of a simple interpretation of RDF semantics [32], where each property is associated not only with a truth extension but also with a falsity extension.

Definition 3 (Partial interpretation). A *partial interpretation* I of a vocabulary V consists of:

- 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[9]: $PT_I : Prop_I \rightarrow \mathcal{P}(Res_I \times Res_I)$.
- 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) $I(x) = IL_I(x)$, $\forall\, x \in V \cap \mathcal{TL}$. □

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. The interpretation of URIs, plain literals and typed literals is done as in the original RDFS semantics.

Example 3. Let a vocabulary $V = \{ex{:}Carlos,\ ex{:}Grigoris,\ ex{:}Riesling,\ ex{:}likes,\ ex{:}denotationOf,\ \ Grigoris\ \hat{}\hat{}xsd{:}string\}$ and consider a structure I that consists of:

- A set of resources $Res_I = \{C, G, R, l, d,\ \ Grigoris\ \}$.
- 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: $PT_I(d) = \{\langle\ \ Grigoris\ \ , G\rangle\}$.
- 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: $IL_I(\ \ Grigoris\ \hat{}\hat{}xsd{:}string) =\ \ Grigoris\ \ .$
- A set of literal values $LV_I = \{\ \ Grigoris\ \ \}$.

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*. □

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

A partial interpretation I of a vocabulary V is *coherent* iff for all $x \in Prop_I$, $PT_I(x) \cap PF_I(x) = \emptyset$, meaning that any triple cannot be simultaneously true and false.

In order to be able to interpret variables in ERDF graphs and formulas, we require as usual the notion of valuations. Let I be a partial interpretation of a vocabulary V and let v be a partial function $v : Var \to Res_I$ (called *valuation*). If $x \in Var$, we define $[I + v](x) = v(x)$. If $x \in V$, we define $[I + v](x) = I(x)$, i.e. we interpret a variable according to the valuation and the remaining vocabulary using the partial interpretation I. We now have all the necessary formal ingredients to define satisfaction of ERDF formulas.

Definition 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 $v : Var(F) \to Res_I$.

- 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 a mapping $u : Var(G) \to Res_I$ such that $u(y) = v(y)$, $\forall y \in Var(G) - \{x\}$, and $I, u \models G$.
- If $F = \forall x\ G$ then $I, v \models F$ iff for all mappings $u : Var(G) \to Res_I$ such that $u(y) = v(y)$, $\forall y \in Var(G) - \{x\}$, it is the case that $I, u \models G$.
- All other cases of ERDF formulas are treated by the following DeMorgan-style rewrite rules expressing the falsification of compound ERDF formulas: $\neg(F \wedge G) \to \neg F \vee \neg G$, $\neg(F \vee G) \to \neg F \wedge \neg G$, $\neg(\neg F) \to F$, $\neg({\sim} F) \to F$[10], $\neg(\exists x\ F) \to \forall x\ \neg F$, $\neg(\forall x\ F) \to \exists x\ \neg F$, $\neg(F \supset G) \to F \wedge \neg G$. □

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 $v : Var(F) \to Res_I$, it is the case 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 is the case 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 [32]. The vocabulary of RDFS, $\mathcal{V}_{RDFS}$, is a set of $\mathcal{URI}$ references in the *rdfs*: namespace [32]. 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 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 5 (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

[10] This transformation expresses that if it is *false* that F *does not hold* then F *holds*.

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 [32], 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 interpretations [32] 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 $\langle x, y\rangle \in PT_I(I(p))$ nor $\langle x, y\rangle \in PF_I(I(p))$. 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 5 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*.

In order to be able to define an appropriate semantics for the negation(s) of ERDF we need to restrict to Herbrand like interpretations, in order to simplify matters. The *vocabulary* of an ERDF ontology $O = \langle G, P\rangle$ 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 (i.e. the set of Herbrand resources Res_O^H is V_O with the well-typed XML literals substituted by their corresponding XML values).

Definition 6 (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 $\mathcal{I}^H(O)$. □

Note that we first defined the notion of an ERDF interpretation and then the notion of a Herbrand interpretation of an ERDF ontology in order to faithfully extend the RDFS semantics [32]. As usual in the construction of Herbrand interpretations, we map every constant to itself except for the predefined XML literals that have a fixed interpretation in the original RDFS semantics. Accordingly, an ERDF Herbrand interpretation can be succinctly described by the property-truth and property-false extensions of properties.

Before we introduce the notion of ERDF stable models, we require some extra definitions. Since we allow arbitrary formulas in the body of rules some extra complexity is necessary to obtain an intuitive semantics in the spirit of partial logic [33]. This is not just a technical exercise and will provide extra expressive power, as we will show later on. 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 $\mathcal{I} \subseteq \mathcal{I}^H(O)$. We define $minimal(\mathcal{I}) = \{I \in \mathcal{I} \mid \nexists J \in \mathcal{I} :\ J \neq I \text{ and } J \leq I\}$. Let $I, J \in \mathcal{I}^H(O)$. We define $[I, J]_O = \{I' \in \mathcal{I}^H(O) \mid I \leq I' \leq J\}$.

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 $v : FVar(r) \to V$. 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 [33].

Definition 7 (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 $I_k = I_{k+1} = M$[11] and:

[11] The condition $I_k = I_{k+1} = M$ actually states that two successive iterations, one computing I_k and the next computing I_{k+1} result to the same Herbrand interpretation of O, which is equal to M.

1. $I_0 \in minimal(\{I \in \mathcal{I}^H(O) \mid I \models sk(G)\})$.
2. For natural numbers α with $0 < \alpha \leq k+1$:
   $I_\alpha \in minimal(\{I \in \mathcal{I}^H(O) \mid I \geq I_{\alpha-1}$ and it is the case that:
   $\forall\, r \in [P]_{V_O}$, if $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O$ then $I \models Concl(r)\})$.

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 a stable model M. In other 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[12] (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.

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

Example 4. Consider a class *HappyParent* whose instances are persons that have children and every child that they have is married and has his/her own children. Additionally, consider the property *isParentOf*(X, Y), indicating that person X is parent to person Y and the property *isMarriedTo*(X, Y) indicating that person X is married to person Y. An ERDF program P that describes this case is the following:

rdf:*type*($?x$, *HappyParent*) $\leftarrow$ *isParentOf*($?x, ?y$),
$\forall ?y$ *isParentOf*($?x, ?y$) $\supset$ ($\exists ?z$ *isMarriedTo*($?y, ?z$), $\exists ?w$ *isParentOf*($?y, ?w$)).

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

$G = \{$ *isParentOf*(*Peter*, *John*), *isParentOf*(*John*, *Mary*), *isMarriedTo*(*John*, *Anastasia*)$\}$.

Then, according to Definition 7, the ERDF ontology $O = \langle G, P\rangle$ has a single stable model, M, such that:

$M \models$ *rdf*:*type*(*Peter*, *HappyParent*) $\wedge$ $\sim$*rdf*:*type*(*John*, *HappyParent*) $\wedge$
$\sim$*rdf*:*type*(*Mary*, *HappyParent*) $\wedge$ $\sim$*rdf*:*type*(*Anastasia*, *HappyParent*).

Stable model M is reached through the chain $I_0 \leq M$, 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:

[12] A sequence of ERDF rules r_1,..., r_n is called *acceptable* if the condition of each rule r_i, for $i \in \{1, ..., n\}$, is satisfied according to the conclusions of the previous rules, which remain applicable throughout the generation of a stable model.

$\{r \in [P]_{V_O} \mid J \models Cond(r), \forall J \in [I_0, M]_O\} =$
$\{r \in [P]_{V_O} \mid J \models Cond(r), \forall J \in [M, M]_O\} =$
$\{rdf{:}type(Peter, HappyParent) \leftarrow isParentOf(Peter, John),$
$\quad \forall ?y\ isParentOf(Peter, ?y) \supset (\exists ?z\ isMarriedTo(?y, ?z),\ \exists ?w\ isParentOf(?y, ?w))\}.$

□

Example 5. Consider the ERDF ontology $O = \langle G, P\rangle$ of Example 1. Then, O has two kinds of stable models $\mathcal{M}_1$ and $\mathcal{M}_2$, where for all $M \in \mathcal{M}_1$, $M \models \neg rdf{:}type(Anne, Adult)$ and for all $M \in \mathcal{M}_2$, $M \models rdf{:}type(Anne, Adult)$. This is because: (i) due to rule (3) of P, $ex{:}Adult$ is a total class and (ii) we do not know if Anne is an adult. For all $M \in \mathcal{M}_1 \cup \mathcal{M}_2$, it holds $M \models \neg rdf{:}type(Anne, Child)$. This is due to the CWA expressed by rule (4) of P. Additionally, for all $M \in \mathcal{M}_1 \cup \mathcal{M}_2$, it is the case that $M \models serveSoftDrink(Anne, Coca\text{-}Cola)$. This is because, if *Anne* is not an adult then, since she is not a child, it is decided to drink *Coca-Cola* (rule (5)). If *Anne* is an adult then, since it is not known if she likes wine, it is also decided to drink *Coca-Cola* (rule (6)). Thus, it is the case that $O \models^{st} \neg rdf{:}type(Anne, Child) \wedge\ serveSoftDrink(Anne, Coca\text{-}Cola)$. Additionally, for all $M \in \mathcal{M}_1 \cup \mathcal{M}_2$, 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) *Carlos* likes *only Retsina*. Thus, it holds $O \models^{st} rdf{:}type(Retsina, SelectedWine) \wedge \sim rdf{:}type(Riesling, SelectedWine)$. □

In [6], 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 that: $G \models^{RDFS} G'$ iff $\langle G, \emptyset\rangle \models^{st} G'$. □

4 Undecidability of ERDF Stable Model Semantics

Unfortunately, satisfiability and entailment under the ERDF stable model semantics are in general undecidable [6]. The proof of undecidability exploits a reduction from the *unbounded tiling problem*, for which existence of a solution is known to be undecidable [11]. 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.

The reduction in [6] 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 is a conjunction of ERDF literals.

Below, we prove a new result also by a reduction from the unbounded tiling problem [11] 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.

The unbounded tiling problem consists of 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 $\tau : \mathbb{N} \times \mathbb{N} \rightarrow \mathcal{T}$ such that: $(\tau(i, j), \tau(i + 1, j)) \in H$ and $(\tau(i, j), \tau(i, j + 1)) \in V$, for all $i, j \in \mathbb{N}$. The existence of a solution for a given unbounded tiling problem is known to be undecidable [11].

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 $O_\mathcal{D} = \langle G_\mathcal{D}, P_\mathcal{D} \rangle$ 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 $ofType(x, T)$, indicating that resource x is of type T, (iii) a class *TileType* whose instances are the types of the tiles, (iii) a class *HasTileType* whose instances are the tiles which have as type an instance of *TileType*, (iv) a property $right(x, y)$ indicating that resource y is right next to resource x, (v) a property $above(x, y)$ indicating that resource y is exactly above resource x, (vi) a class *HasTileRight* whose instances are the tiles for which there exists a tile right next to them, (vii) a class *HasTileAbove* whose instances are the tiles for which there exists a tile exactly above them, (viii) a property $HConstraint(T, T')$, indicating that $(T, T') \in H$, and (ix) a property $VConstraint(T, T')$, indicating that $(T, T') \in V$.

Let $G_\mathcal{D}$ be the ERDF graph:

$$
\begin{aligned}
G_\mathcal{D} = \; & \{rdfs{:}subClassOf(rdfs{:}ContainerMembershipProperty, Tile), \\
& \; rdfs{:}subClassOf(Tile, rdfs{:}ContainerMembershipProperty)\} \cup \\
& \{rdf{:}type(T, TileType) \mid T \in \mathcal{T}\} \cup \\
& \{HConstraint(T, T') \mid (T, T') \in H\} \cup \{VConstraint(T, T') \mid (T, T') \in V\} \cup \\
& \{rdf{:}type(ofType, erdf{:}TotalProperty), rdf{:}type(right, erdf{:}TotalProperty), \\
& \; rdf{:}type(above, erdf{:}TotalProperty)\}.
\end{aligned}
$$

Let $P_\mathcal{D}$ be the ERDF program, containing the following rules:

$$
\begin{aligned}
rdf{:}type(?x, HasTileType) \leftarrow \; & rdf{:}type(?x, Tile) \wedge rdf{:}type(?y, TileType) \wedge \\
& ofType(?x, ?y).
\end{aligned}
$$

$$
\begin{aligned}
rdf{:}type(?x, HasTileAbove) \leftarrow \; & rdf{:}type(?x, Tile) \wedge rdf{:}type(?y, Tile) \wedge \\
& above(?x, ?y).
\end{aligned}
$$

$$
\begin{aligned}
rdf{:}type(?x, HasTileRight) \leftarrow \; & rdf{:}type(?x, Tile) \wedge rdf{:}type(?y, Tile) \wedge \\
& right(?x, ?y).
\end{aligned}
$$

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

Note that in all stable models of $O_\mathcal{D} = \langle G_\mathcal{D}, P_\mathcal{D} \rangle$, the class *Tile* contains exactly the (infinite in number) $rdf{:}\_i$ terms, for $i \in \mathbb{N}$. This is because, computing the stable models of O, only the minimal models of $sk(G)$ are considered (see Definition 7, Step 1). Thus, each tile on the infinite grid is represented by an $rdf{:}\_i$ term, for $i \in \mathbb{N}$.

To finalize the reduction, we define:

$$
\begin{aligned}
F_\mathcal{D} = (\exists ?x \exists ?y \exists ?x' \exists ?y' \exists ?x'' \quad & rdf{:}type(?x, Tile) \wedge \; rdf{:}type(?y, Tile) \wedge \\
& rdf{:}type(?x', Tile) \wedge rdf{:}type(?y', Tile) \wedge \\
& rdf{:}type(?x'', Tile) \wedge right(?x, ?y) \wedge \\
& above(?y, ?y') \wedge right(?x', ?y') \wedge \\
& above(?x'', ?x') \wedge {\sim}id(?x, ?x'')) \vee
\end{aligned}
$$

$$
(\exists ?x \quad rdf{:}type(?x, Tile) \wedge {\sim}rdf{:}type(?x, HasTileType)) \vee
$$

$$(\exists ?x \exists ?T \exists ?T' \quad rdf{:}type(?x, Tile) \wedge rdf{:}type(?T, TileType) \wedge rdf{:}type(?T', TileType) \wedge ofType(?x, ?T) \wedge ofType(?x, ?T') \wedge \sim id(?T, ?T')) \vee$$

$$(\exists ?x \quad rdf{:}type(?x, Tile) \wedge \sim rdf{:}type(?x, HasTileRight)) \vee$$

$$(\exists ?x \exists ?y \exists ?y' \quad rdf{:}type(?x, Tile) \wedge rdf{:}type(?y, Tile) \wedge rdf{:}type(?y', Tile) \wedge right(?x, ?y) \wedge right(?x, ?y') \wedge \sim id(?y, ?y')) \vee$$

$$(\exists ?x \quad rdf{:}type(?x, Tile) \wedge \sim rdf{:}type(?x, HasTileAbove)) \vee$$

$$(\exists ?x \exists ?y \exists ?y' \quad rdf{:}type(?x, Tile) \wedge rdf{:}type(?y, Tile) \wedge rdf{:}type(?y', Tile) \wedge above(?x, ?y) \wedge above(?x, ?y') \wedge \sim id(?y, ?y')) \vee$$

$$(\exists ?x \exists ?y \exists ?T \exists ?T' \quad rdf{:}type(?x, Tile) \wedge rdf{:}type(?y, Tile) \wedge rdf{:}type(T, TileType) \wedge rdf{:}type(T', TileType) \wedge ofType(?x, ?T) \wedge ofType(?y, ?T') \wedge right(?x, ?y) \wedge \sim HConstraint(?T, ?T')) \vee$$

$$(\exists ?x \exists ?y \exists ?T \exists ?T' \quad rdf{:}type(?x, Tile) \wedge rdf{:}type(?y, Tile) \wedge rdf{:}type(T, TileType) \wedge rdf{:}type(T', TileType) \wedge ofType(?x, ?T) \wedge ofType(?y, ?T') \wedge above(?x, ?y) \wedge \sim VConstraint(?T, ?T'))$$

Formula $F_{\mathcal{D}}$ expresses that:

1. there is a tile x such that, starting from x, if we move:

   $$one\ tile\ right \rightarrow one\ tile\ up \rightarrow one\ tile\ left \rightarrow one\ tile\ down$$

   then we will meet a tile x'' different than x, or
2. there is a tile that is not associated with exactly one tile type, or
3. there is a tile x that does not have exactly one tile right next to it, or
4. there is a tile x that does not have exactly one tile right above it, or
5. the types of the horizontally adjacent tiles do not respect the H relation of $\mathcal{D}$, or
6. the types of the vertically adjacent tiles do not respect the V relation of $\mathcal{D}$.

Proposition 2. Let $\mathcal{D}$ be an instance of the unbounded tiling problem. It holds: $\mathcal{D}$ has a solution iff $O_{\mathcal{D}} \not\models^{st} F_{\mathcal{D}}$. □

From the previous proposition, it follows directly the following one.

Proposition 3. Let O be an objective ERDF ontology and let F be a general ERDF formula. The problem $O \models^{st} F$ is undecidable. □

In [6], the undecidability result of the ERDF stable model semantics was achieved by having rules in the corresponding ERDF program containing weak and strong negation that guarantee that tiles are placed (i) one next to the other and (ii) one above to the other satisfying the horizontal and vertical tile adjacency constraints. Now the correct placement of tiles is expressed in the entailed ERDF formula $F_{\mathcal{D}}$.

5 #*n*-Stable Model Semantics

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, because of the container membership properties. Thus, the vocabulary of O is also infinite (note that $\{rdf{:}\_i \mid i \geq 1\} \subseteq \mathcal{V}_{RDF} \subseteq V_O$). Consequently, 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 by limiting the maximum number of container membership properties. 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. 5 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 $Res_O^{H_{\#n}} = Res_O^H - \{rdf{:}\_i \mid i > n\}$.

Recall that the $rdf{:}\_i$ properties are used to express members of containers (i.e. bags, sequences, and alternatives), which are in practice finitely limited. Since undecidability is achieved even with the presence of weak and strong negation and the presence of the infinite number of $rdf{:}\_i$ terms as shown in [6], it seems reasonable to make the number of $rdf{:}\_i$ terms finite and keep as many as you need.

This has a parallel in the OWL 2 language, where two semantics are defined [34]: the OWL 2 Direct Semantics (or OWL 2 DL), and the OWL 2 RDF-Based Semantics (or OWL 2 Full). The OWL 2 Full semantics is the natural extension of RDFS semantics, but it is undecidable. The OWL 2 Direct Semantics is decidable, and is a restriction of OWL 2 Full. We have followed the same approach in this paper.

The changes are straightforward and obtained by restricting to the $V_O^{\#n}$ vocabulary in the previous definitions.

Definition 8 (#*n*-Herbrand interpretation). 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 $I, J \in \mathcal{I}^{H_{\#n}}(O)$. We define $[I, J]_O^{\#n} = \{I' \in \mathcal{I}^{H_{\#n}}(O) \mid I \leq I' \leq J\}$.

Definition 9 (ERDF #*n*-stable model). Let $O = \langle G, P \rangle$ be an ERDF ontology and let $M \in \mathcal{I}^{H_{\#n}}(O)$. We say that M is an *(ERDF) #n-stable model* of O iff there is a chain of #n-Herbrand interpretations of O, $I_0 \leq ... \leq I_{k+1}$, such that $I_k = I_{k+1} = M$ and:

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

2. For natural numbers α with $0 < \alpha \leq k+1$:
   $I_\alpha \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \geq I_{\alpha-1}$ and it is the case that:
   $\forall\, r \in [P]_{V_O^{\#n}}$, if $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}$, then $I \models Concl(r)\})$.

The set of $\#n$-stable models of O is denoted by $\mathcal{M}^{st_{\#n}}(O)$. □

Below we define entailment of an ERDF formula or ERDF graph from an ERDF ontology.

Definition 10 (ERDF formula or ERDF graph entailment). 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 $M \in \mathcal{M}^{st_{\#n}}(O)$, $M \models F$. □

The answers (solutions) to a query represented by an ERDF formula F over an ontology O are in fact variable mappings of the free variables to the vocabulary for which the query holds. This is exactly the approach followed in SPARQL semantics [59] to represent solutions to queries. Formally, 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[13]:

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

Notice that if there are no free variables then the query has a simple “yes” or “no” response, which can be used to capture `ASK` queries of the SPARQL language [60] over ordinary RDF graphs. When a variable mapping is returned, the above definition generalizes SPARQL semantics for RDF graphs since in SPARQL only one model has to be considered while in our case several models may have to be taken into account (see Example 7 below).

Example 6. Consider the ERDF ontology O of Example 4. Then, similarly to Example 4, O has one $\#n$-stable model M, for $n \geq 1$, such that:

$$M \models\ rdf{:}type(Peter, HappyParent)\ \wedge\ {\sim}rdf{:}type(John, HappyParent)\ \wedge\ {\sim}rdf{:}type(Mary, HappyParent)\ \wedge {\sim}rdf{:}type(Anastasia, HappyParent).$$

Thus,

$$O \models^{st_{\#n}} rdf{:}type(Peter, HappyParent)\ \wedge\ {\sim}rdf{:}type(John, HappyParent)\ \wedge\ {\sim}rdf{:}type(Mary, HappyParent)\ \wedge {\sim}rdf{:}type(Anastasia, HappyParent).$$ □

Example 7. Consider the ERDF ontology O of Example 1. Then, similarly to Example 5, O has two kinds of $\#n$-stable models, for an $n \geq 1$, $\mathcal{M}_1$ and $\mathcal{M}_2$, for which hold the same results as these described in Example 5. Thus, it is the case that $O \models^{st_{\#n}} {\neg}rdf{:}type(Anne, Child)\ \wedge serveSoftDrink(Anne, Coca\text{-}Cola)$. Additionally, $O \models^{st_{\#n}} rdf{:}type(Retsina, SelectedWine)\ \wedge {\sim}rdf{:}type(Riesling, SelectedWine)$. □

Theorem 1 below relates (original) stable model entailment and $\#n$-stable model entailment. First, we provide a definition for an important subset of ERDF formulas.

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

Definition 11 (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) $FVar(F) = \emptyset$. □

For example, let:

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

Then, F is an ERDF d-formula. It is easy to see that if G is an ERDF graph then $\mathit{formula}(G)$ is an ERDF d-formula. Since there are no free variables, d-formulas have a boolean "yes" or "no" answer.

The following proposition states that stable model entailment of d-formulas from objective ERDF ontologies coincides with $\#n$-stable model entailment, provided that n is large enough to include all $\mathit{rdf{:}\_i}$ terms appearing in the queried ontology. The concrete value n is determined resorting to the following auxiliary definition. Let $O = \langle G, P\rangle$ be an ERDF ontology. We define:

$$n_O = \begin{cases} 1, & \text{if } (V_G \cup V_P) \cap \{\mathit{rdf{:}\_i} \mid i \geq 1\} = \emptyset \\ \mathit{max}(\{i \in I\!N \mid \mathit{rdf{:}\_i} \in V_G \cup V_P\}), & \text{otherwise} \end{cases}$$

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

Theorem 1. Let O be an objective ERDF ontology and let $n \geq n_O$. Let F^d be an ERDF d-formula s.t. $\mathit{max}(\{i \in I\!N \mid \mathit{rdf{:}\_i} \in V_{F^d}\}) \leq n$. It holds: $O \models^{st} F^d$ iff $O \models^{st_{\#n}} F^d$. □

Since $V_O^{\#n}$ (for $n \in I\!N$) 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, Theorem 1 does not hold in the case that $O = \langle G, P\rangle$ is a general ERDF ontology. Moreover, Theorem 1 does not hold when F^d is a general ERDF formula. For example, consider the ERDF graph G:

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

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

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

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

It follows, from Theorem 1, the following Corollary.

Corollary 1. Let $O = \langle G, P\rangle$ be an objective ERDF ontology and let $n \geq n_O$. It is the case that: O has a stable model iff O has a $\#n$-stable model. □

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

Proposition 4. Let G, G' be RDF graphs such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$, $V_{G'} \cap \mathcal{V}_{ERDF} = \emptyset$, and $V_{G'} \cap sk_G(\mathit{Var}(G)) = \emptyset$. Let $O = \langle G, \emptyset\rangle$ and $n \geq n_O$. If $\mathit{max}(\{i \in I\!N \mid \mathit{rdf{:}\_i} \in V_{G'}\}) \leq n$ then: $G \models^{RDFS} G'$ iff $O \models^{st_{\#n}} G'$. □

Existing RDFS engines like [37, 17] take care at runtime to avoid the infinite instantiation of container membership properties, guaranteeing termination of their algorithms. Our results above formalize this process for our semantics, guaranteeing decidabiltiy, soundness and completeness for important subsets of queries, namely for performing standard RDFS entailment between graphs, as stated in Proposition 4.

6 Query Answering on Simple ERDF Ontologies

In this section, we consider another decidable fragment of ERDF ontologies where the program rules are simple, i.e. the bodies are formed by conjunctions of ERDF triples and weak negations of ERDF triples. Technically, we will show that the $\#n$-stable answers of a simple ERDF formula F with respect to a simple ERDF ontology $O = \langle G, P \rangle$ can be computed through Answer Set Programming [29, 30][14] on an extended logic program (ELP) $\Pi_O^{\#n}$ constructed from the ontology O. By using an answer set solver like `smodels` [57], `dlv` [42], or `clasp` [26], one can perform query answering on simple ERDF ontologies.

To give the precise definition of $\Pi_O^{\#n}$, a few auxiliary definitions are needed. Let t be an ERDF triple, *true*, or *false*. We define:

$$L_t = \begin{cases} H(s,p,o) & \text{if } t = p(s,o) \\ \neg H(s,p,o) & \text{if } t = \neg p(s,o) \\ t & \text{if } t \in \{true,\ false\} \end{cases}$$

Intuitively, L_t is an extended logic programming literal (ELP literal[15]), representing the ERDF triple t.

Let G be an ERDF graph. We define: $\Pi_G = \{L_t \leftarrow true \mid t \in sk(G)\}$. Intuitively, Π_G is a set of ELP facts, representing $sk(G)$. For example, let $G = \{hasMother(Manos, Anastasia)\}$. Then, $\Pi_G = \{H(Manos, hasMother, Anastasia) \leftarrow true\}$.

Let P be a simple ERDF program. We define:

$$\Pi_P = \{L_{t_0} \leftarrow L_{t_1}, ..., L_{t_k}, {\sim}L_{t_{k+1}}, ..., {\sim}L_{t_m} \mid\ t_0 \leftarrow t_1, ..., t_k, {\sim}t_{k+1}, ..., {\sim}t_m \in P\}.$$

Intuitively, Π_P is an extended logic program, representing P. For example, let $P = \{hasChild(?y, ?x) \leftarrow hasMother(?x, ?y)\}$. Then, $\Pi_P = \{H(?y, hasChild, ?x) \leftarrow H(?x, hasMother, ?y)\}$.

Let O be an ERDF ontology. We denote by $\Pi_O^{H_{\#n}}$ the extended logic program that consists of the following two sets of rules[16]:

Partial Interpretation Rules

$H(?z, type, Property) \leftarrow H(?x, ?z, ?y).$
$H(?z, type, Property) \leftarrow \neg H(?x, ?z, ?y).$ (+)
For all $x \in V_O^{\#n}$: $H(x, type, Resource) \leftarrow true.$
For all $x \in V_O^{\#n} \cap \mathcal{PL}$: $H(x, type, Literal) \leftarrow true.$

[14] Other references of Answer Set Programming include [49, 56, 45, 9, 46, 14, 25].
[15] By *ELP literal*, we refer to an atom or the strong negation of an atom.
[16] For simplicity, we have eliminated the namespace from the URIs in $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$.

ERDF Interpretation Rules

$H(?z, type, ?y) \leftarrow H(?x, domain, ?y), H(?z, ?x, ?w).$
$H(?w, type, ?y) \leftarrow H(?x, range, ?y), H(?z, ?x, ?w).$

$H(?x, subClassOf, Resource) \leftarrow H(?x, type, Class).$

$H(?x, type, Class) \leftarrow H(?x, subClassOf, ?y).$
$H(?y, type, Class) \leftarrow H(?x, subClassOf, ?y).$
$H(?z, type, ?y) \leftarrow H(?x, subClassOf, ?y), H(?z, type, ?x).$
$\neg H(?z, type, ?x) \leftarrow H(?x, subClassOf, ?y), \neg H(?z, type, ?y).$ (†)
$H(?x, subClassOf, ?x) \leftarrow H(?x, type, Class).$
$H(?x, subClassOf, ?z) \leftarrow H(?x, subClassOf, ?y), H(?y, subClassOf, ?z).$

$H(?x, type, Property) \leftarrow H(?x, subPropertyOf, ?y).$
$H(?y, type, Property) \leftarrow H(?x, subPropertyOf, ?y).$
$H(?z_1, ?y, ?z_2) \leftarrow H(?x, subPropertyOf, ?y), H(?z_1, ?x, ?z_2).$
$\neg H(?z_1, ?x, ?z_2) \leftarrow H(?x, subPropertyOf, ?y), \neg H(?z_1, ?y, ?z_2).$ (†)
$H(?x, subPropertyOf, ?x) \leftarrow H(?x, type, Property).$
$H(?x, subPropertyOf, ?z) \leftarrow H(?x, subPropertyOf, ?y), H(?y, subPropertyOf, ?z).$

$H(?x, subClassOf, Literal) \leftarrow H(?x, type, Datatype).$
$H(?x, subPropertyOf, member) \leftarrow H(?x, type, ContainerMembershipProperty).$

$\neg H(?z, type, ?x) \leftarrow H(?x, type, TotalClass), {\sim}H(?z, type, ?x).$ ($\star$)
$H(?z, type, ?x) \leftarrow H(?x, type, TotalClass), {\sim} \neg H(?z, type, ?x).$ ($\star$)

$\neg H(?z_1, ?x, ?z_2) \leftarrow H(?x, type, TotalProperty), {\sim}H(?z_1, ?x, ?z_2).$ ($\star$)
$H(?z_1, ?x, ?z_2) \leftarrow H(?x, type, TotalProperty), {\sim} \neg H(?z_1, ?x, ?z_2).$ ($\star$)

For each $s \text{ ^^}rdf{:}XMLLiteral \in V_O^{\#n}$ s.t. s is a well-typed XML literal string:
$H(\ s \text{ ^^}rdf{:}XMLLiteral, type, XMLLiteral) \leftarrow true.$

For each $s \text{ ^^}rdf{:}XMLLiteral \in V_O^{\#n}$ s.t. s is not a well-typed XML literal string:
$\neg H(\ s \text{ ^^}rdf{:}XMLLiteral, type, Literal) \leftarrow true.$

For each RDF, RDFS, or ERDF axiomatic triple $p(s, o)$ s.t. $p, s, o \in V_O^{\#n}$:
$H(s, p, o) \leftarrow true.$

Intuitively, the Partial Interpretation Rules of an ERDF ontology O represent the semantic conditions in Def. 3 (*Partial Interpretation*). Similarly, the ERDF Interpretation Rules represent the semantic conditions in Def. 5 (*ERDF Interpretation*) that a $\#n$-Herbrand interpretation of O (for $n \in \mathbb{N}$) satisfies. For example, the rules marked with (†) capture reasoning with the property-false extensions, specific of ERDF. Additionally, note that the ERDF Interpretation Rules, corresponding to metaclasses *erdf*:*TotalClass* and *erdf*:*TotalProperty* (indicated by ($\star$)), represent the semantic conditions 12 and 13 of Def. 5, respectively. Due to these rules, the OWA applies to the truth values of *rdf*:*type*(x, c) and $p(x, y)$, for total classes c and total properties p. By dropping rules marked with (+), (†), and ($\star$), we obtain complete inference rules for RDFS reasoning similar to [68]. Note that similarly to [68], we achieve completeness of RDFS reasoning, by allowing variables in the property position of the atom $H(subject, property, object)$.

The intended extended logic program is obtained by translating the triples in the ERDF graph, the ERDF program rules, and adding the previous Partial and ERDF

interpretation rules. Formally, let $O = \langle G, P \rangle$ be a simple ERDF ontology and let $n \in \mathbb{N}$. We define:

$$\Pi_O^{\#n} = \Pi_G \cup \Pi_P \cup \Pi_O^{H_{\#n}}.$$

To proceed in computing the stable models of an ERDF ontology, we need the following auxiliary definition. Intuitively, a $\#n$-semi-Herbrand interpretation of an ERDF ontology O is a coherent, partial interpretation that satisfies the three conditions of a $\#n$-Herbrand interpretation regarding the interpretation of the vocabulary, while the rest of the conditions are just the definitions of the ontological categories present in the definition of ERDF interpretation. The definition of a $\#n$-semi-Herbrand interpretation is needed for mapping $\#n$-Herbrand interpretations to consistent sets of ELP literals, and vice-versa.

Definition 12 ($\#n$-semi-Herbrand interpretation). Let O be an ERDF ontology. A *$\#n$-semi-Herbrand interpretation* I of O is a coherent, partial interpretation of $V_O^{\#n}$, extended by the 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:

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 $V_O^{\#n}$.

Additionally:

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:
   $Prop_I = CT_I(rdf{:}Property)$ $\quad Cls_I = CT_I(rdfs{:}Class)$
   $Res_I = CT_I(rdfs{:}Resource)$ $\quad LV_I = CT_I(rdfs{:}Literal)$
   $TCls_I = CT_I(erdf{:}TotalClass)$ $\quad TProp_I = CT_I(erdf{:}TotalProperty)$. □

Let O be an ERDF ontology. Below, we show how a $\#n$-semi-Herbrand interpretation I of O can be translated to a consistent Herbrand interpretation of $\Pi_O^{\#n}$, denoted by $ELP(I)$. Let I be a $\#n$-semi-Herbrand interpretation of O. We define:

$$\begin{aligned} ELP(I) = \; & \{H(s,p,o) \mid s,p,o \in V_O^{\#n} \text{ and } \langle I(s), I(o) \rangle \in PT_I(p)\} \cup \\ & \{\neg H(s,p,o) \mid s,p,o \in V_O^{\#n} \text{ and } \langle I(s), I(o) \rangle \in PF_I(p)\} \end{aligned}$$

Note that the function $ELP(.)$ from the set of $\#n$-semi-Herbrand interpretations of O to the set of consistent sets of ELP literals $H(s,p,o)$ and $\neg H(s,p,o)$, where $s,p,o \in V_O^{\#n}$, is a bijective mapping.

Theorem 2. Let O be a simple ERDF ontology and let $n \geq n_O$. Let M be a $\#n$-semi-Herbrand interpretation of O. It is the case that: $M \in \mathcal{M}^{st_{\#n}}(O)$ iff $ELP(M)$ is a consistent answer set of $\Pi_O^{\#n}$. □

This result captures the intuitive idea that conditions (3-17) of ERDF interpretations can be captured by extended logic programming rules under answer set semantics, for the particular case of simple ontologies. Based on Theorem 2, the set $S = \{ELP^{-1}(N) \mid N \text{ is a consistent answer set of } \Pi_O^{\#n}\}$, for $n \geq n_O$, computes the set of all $\#n$-stable models of O.

Using again Theorem 2, we can show that the $\#n$-stable answers of a simple ERDF formula F w.r.t. a simple ERDF ontology O can be computed through Answer Set Programming [29, 30] on $\Pi_O^{\#n}$. First, we provide a few definitions. 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 [30]. Let F be a simple ERDF formula of the form: $F = t_1 \wedge ... \wedge t_k \wedge \sim t_{k+1} \wedge ... \wedge \sim t_n$, where each $t_i = [\neg]p_i(s_i, o_i)$. We define:

$$L_F = L_{t_1} \wedge \cdots \wedge L_{t_k} \wedge \sim L_{t_{k+1}} \wedge \cdots \wedge \sim L_{t_n}.$$

Proposition 5. Let O be a simple ERDF ontology and let $n \geq n_O$. Let F be a simple ERDF formula over $V_O^{\#n}$.

1. If $\Pi_O^{\#n}$ is a non-contradictory ELP then $Ans_O^{st_{\#n}}(F) = Ans_{\Pi_O^{\#n}}^{AS}(L_F)$.
2. Otherwise, $\mathcal{M}^{st_{\#n}}(O) = \emptyset$. □

Note that we represent an ERDF triple $[\neg]p(s, o)$ by the ELP literal $[\neg]H(s, p, o)$, and not by the ELP literal $[\neg]p(s, o)$. This is because according to RDF(S) semantics [32] and thus ERDF $\#n$-stable model semantics, all properties, classes, and instances are resources on which inferences can be made. Neglecting this fact results in incompleteness, as what happens with the RDFS entailment rules present in the RDFS recommendation [68]. For example, consider the following rules of $\Pi_O^{H_{\#n}}$:

$$\begin{aligned}
H(?z_1, ?y, ?z_2) &\leftarrow H(?x, \mathit{subPropertyOf}, ?y), H(?z_1, ?x, ?z_2).\\
\neg H(?z_1, ?x, ?z_2) &\leftarrow H(?x, \mathit{subPropertyOf}, ?y), \neg H(?z_1, ?y, ?z_2).\\
\neg H(?z_1, ?x, ?z_2) &\leftarrow H(?x, \mathit{type}, \mathit{TotalProperty}), \sim H(?z_1, ?x, ?z_2).\\
H(?z_1, ?x, ?z_2) &\leftarrow H(?x, \mathit{type}, \mathit{TotalProperty}), \sim \neg H(?z_1, ?x, ?z_2).
\end{aligned}$$

Note that in the head of the rules and in the second ELP literal in the bodies of the rules, there is a variable in the property position p of the corresponding ELP literal $[\neg]H(s, p, o)$. Additionally, all variables of the first ELP literal in the body of the rules correspond to properties. In some sense, limited high-order features are necessary to fully capture RDFS reasoning which are not available in ordinary extended logic programming. Generalizations like HiLog [16] are necessary to handle this.

The reader may wonder why we do not define the ERDF $\#n$-stable model semantics of simple ERDF ontologies directly based on the answer set semantics of $\Pi_O^{\#n}$. However, in this case, the definition of the ERDF $\#n$-stable model semantics of simple ERDF ontologies would had been restricted without covering the case of general ERDF ontologies, limiting expressivity as we shall see. In contrast, in our case, we provide a pure model-theoretic definition of the $\#n$-stable model semantics of ERDF ontologies, faithfully extending RDFS semantics. In particular:

– We extend RDF graphs to ERDF graphs with the inclusion of strong negation, and then to ERDF ontologies with the inclusion of general derivation rules.

ERDF graphs allow us 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 $\forall$, $\exists$.
- 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. These terms together with the default closure rules, defined in Section 1, allow the combination of open-world and closed-world reasoning in the same framework. Examples and arguments of this need are provided in [5, 6, 3].
- We extend RDFS interpretations to ERDF interpretations, including both truth and falsity extensions for properties and classes, and providing additional conditions involving falsity extensions and the new metaclasses of total properties and total classes.
- We define the Herbrand interpretations of an ERDF ontology and, based on these, the stable models of an ERDF ontology. Since ERDF stable model satisfiability and entailment are undecidable even for simple ERDF ontologies, we define the ERDF $\#n$-interpretations, the $\#n$-Herbrand interpretations, and the $\#n$-stable model semantics of ERDF ontologies by removing the infinite number of terms *rdf*:_i, where $i > n$. Additionally, we show that $\#n$-stable model entailment on simple ERDF ontologies can be computed through Answer Set Programming on $\Pi_O^{\#n}$.

A comparison of the $\#n$-stable model semantics on general ERDF ontologies with other semantics of logic programs with rules of a richer (than normal programs) syntax is provided at the end of Section 11. All semantics that we consider for this comparison extend the stable model semantics on normal programs [29].

7 Complexity Results for Simple & Objective ERDF Ontologies

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. Recall that reasoning with simple ERDF ontologies is undecidable, and thus we do not address them in this section. We also consider the orthogonal notion of bounded ERDF ontologies

7.1 Bounded ontologies

We consider first the case of bounded ontologies since, as we will show, reasoning tasks will be on the first level of the polynomial hierarchy. As expected, reasoning with bounded ontologies will have lower complexity than reasoning with unbounded ones. To prove this we start from the well-known NP-complete graph coloring problem.

Let $D = (V, E)$ be a graph. We say that D is *3-colorable*, if there is a mapping *color* from the vertices V to colors $\{Red, Green, Blue\}$ such that if $(v, u) \in E$ then $color(v) \neq color(u)$. The *3-colorability problem* is the following: given a graph D, decide if it is 3-colorable. From D, we will construct a bounded, objective ERDF ontology $O_D = \langle G_D, P_D \rangle$ such that D is *3-colorable* iff O_D has a $\#n$-stable model, for $n \in I\!N$. The *3-colorability* problem is a classical NP-complete problem. Thus,

based on this reduction, we will show that the satisfiability problem of a bounded, objective ERDF ontology, under the $\#n$-stable model semantics is NP-hard.

Consider (i) a property $edge(v, u)$, indicating that there is an edge in D from vertex v to vertex u, (ii) a class Red whose instances are vertices of color Red, (iii) a class $Green$ whose instances are vertices of color $Green$, (iv) a class $Blue$ whose instances are vertices of color $Blue$, (v) a property $edgeR(v, u)$, indicating that in edge $\langle v, u\rangle$, the color of v is Red, (vi) a property $edgeG(v, u)$, indicating that in edge $\langle v, u\rangle$, the color of v is $Green$, (vii) a property $edgeB(v, u)$, indicating that in edge $\langle v, u\rangle$, the color of v is $Blue$.

Let G_D be the ERDF graph:

$$\begin{aligned} G_D = \{&edge(v,u) \mid \langle v,u\rangle \in E\} \cup \\ \{&rdf{:}type(edgeR, erdf{:}TotalProperty), rdf{:}type(edgeG, erdf{:}TotalProperty), \\ &rdf{:}type(edgeB, erdf{:}TotalProperty), \\ &rdfs{:}domain(edgeR, Red), rdfs{:}range(edgeR, NotRed), \\ &rdfs{:}domain(edgeG, Green), rdfs{:}range(edgeG, NotGreen), \\ &rdfs{:}domain(edgeB, Blue), rdfs{:}range(edgeB, NotBlue), \\ &rdfs{:}subClassOf(Red, NotGreen), rdfs{:}subClassOf(Red, NotBlue), \\ &rdfs{:}subClassOf(Blue, NotGreen), rdfs{:}subClassOf(Blue, NotRed), \\ &rdfs{:}subClassOf(Green, NotRed), rdfs{:}subClassOf(Green, NotBlue)\} \end{aligned}$$

Let P_D be the bounded objective ERDF program, containing the following constraints:

$$\begin{aligned} \{&false \leftarrow edge(x,y) \wedge \neg edgeR(x,y) \wedge \neg edgeG(x,y) \wedge \neg edgeB(x,y). \\ &false \leftarrow rdf{:}type(x, Red) \wedge rdf{:}type(x, NotRed). \\ &false \leftarrow rdf{:}type(x, Green) \wedge rdf{:}type(x, NotGreen). \\ &false \leftarrow rdf{:}type(x, Blue) \wedge rdf{:}type(x, NotBlue) \mid x, y \in V\} \end{aligned}$$

Proposition 6. Let $D = (V, E)$ be a graph. D is *3-colorable* iff $O_D = \langle G_D, P_D\rangle$ has a $\#n$-stable model, for $n \in \mathbb{N}$. □

Using Proposition 6, we can provide the complexity of the satisfiability problem of a bounded, simple ERDF ontology w.r.t. the ERDF $\#n$-stable model semantics.

First, we give a few definitions that will be used throughout the algorithms and proofs. Let Π be an ELP. By $[\Pi]$, we denote the grounded version of Π and by $EHB(\Pi)$, we denote the *Extended Herbrand Base* of P. Let Π be a ground ELP. Let $r = L_0 \leftarrow L_1, ..., L_m, {\sim}L_{m+1}, ..., {\sim}L_n \in \Pi$. We define: $Head(r) = L_0$, $Body(r)^+ = \{L_1, ..., L_m\}$, $Body(r)^- = \{L_{m+1}, ..., L_n\}$, and $Body(r) = Body(r)^+ \cup Body(r)^-$. Let $N \subseteq EHB(\Pi)$. We define:
$\Pi^N = \{Head(r) \leftarrow Body(r)^+ \mid r \in \Pi$ and $Body(r)^- \cap N = \emptyset\}$.

Let Π be a ground ELP such that for any $r \in \Pi$, it holds $Body(r)^- = \emptyset$. We define the mapping $T_\Pi : \mathcal{P}(EHB(\Pi)) \rightarrow \mathcal{P}(EHB(\Pi))$, where $T_\Pi(N) = N \cup \{Head(r) \mid r \in \Pi$ and $Body(r) \subseteq N\}$.

Proposition 7. Let $O = \langle G, P\rangle$ be a bounded, simple ERDF ontology and let $n \geq n_O$. The problem of establishing whether O has a $\#n$-stable model is NP-complete w.r.t. the size of O. □

Below, we present the Algorithm 5.1 *Satisfies*(I, F) that checks if a partial interpretation I of a vocabulary V satisfies an ERDF formula F with $FVar(F) = \emptyset$.

This algorithm and Proposition 7 will be used to provide the complexity of the query answering problem on a bounded, simple ERDF ontology w.r.t. the $\#n$-stable model semantics.

Algorithm 7.1 *Satisfies*(I, F)

Input: I is a partial interpretation of a vocabulary V and F is an ERDF query s.t. $FVar(F) = \emptyset$
Output: TRUE, if $I \models F$, and FALSE, otherwise

(1) If $V_F \not\subseteq V$ then return(FALSE);
(2) case(F) {
(3) $p(s,o)$: If $p \in \mathcal{URI}$, $p \in Prop_I$, $\langle I(s), I(o)\rangle \in PT_I(I(p))$ then return(TRUE);
(4) $\neg p(s,o)$: If $p \in \mathcal{URI}$, $p \in Prop_I$, $\langle I(s), I(o)\rangle \in PF_I(I(p))$ then return(TRUE);
(5) $\sim G$: If *Satisfies*(I, G)=FALSE then return(TRUE);
(6) $F_1 \wedge F_2$: If *Satisfies*(I, F_1)=TRUE and *Satisfies*(I, F_2)=TRUE then return(TRUE);
(7) $F_1 \vee F_2$: If *Satisfies*(I, F_1)=TRUE or *Satisfies*(I, F_2)=TRUE then return(TRUE);
(8) $F_1 \supset F_2$: return(*Satisfies*$(I, \sim F_1 \vee F_2)$);
(9) $\forall x G$: If for all $u : \{x\} \to V$, it is the case that *Satisfies*$(I, u(G))$=TRUE then return(TRUE);
(10) $\exists x G$: If it exists $u : \{x\} \to V$ such that *Satisfies*$(I, u(G))$=TRUE then return(TRUE);
(11) $\neg(F_1 \wedge F_2)$: return(*Satisfies*$(I, \neg F_1 \vee \neg F_2)$);
(12) $\neg(F_1 \vee F_2)$: return(*Satisfies*$(I, \neg F_1 \wedge \neg F_2)$);
(13) $\neg(\neg G)$: return(*Satisfies*(I, G));
(14) $\neg(\sim G)$: return(*Satisfies*(I, G));
(15) $\neg(\exists x\ G)$: return(*Satisfies*$(I, \forall x\ \neg G)$);
(16) $\neg(\forall x\ G)$: return(*Satisfies*$(I, \exists x\ \neg G)$);
(17) $\neg(F_1 \supset F_2)$: return(*Satisfies*$(I, F_1 \wedge \neg F_2)$);
}
(11) return(FALSE);

Proposition 8. Let $O = \langle G, P\rangle$ be a bounded, simple ERDF ontology and let F be an ERDF formula. Additionally, let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \to V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \geq n_O$. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is co-NP-complete w.r.t. the size of O. □

Below, we state complexity results for the $\#n$-stable model semantics of bounded, objective ERDF ontologies. We see that even though no weak negation appears in the rules of bounded, objective ERDF ontologies, complexity of reasoning remains the same with that of bounded, simple ERDF ontologies. This is due to the ERDF metaclasses *erdf*:*TotalClass* and *erdf*:*TotalProperty*, to the instances of which, the OWA applies. This is natural, since the OWA is captured by rules making use of weak negation generating alternative models.

Proposition 9. Let $O = \langle G, P\rangle$ be a bounded, objective ERDF ontology. Let G' be an ERDF graph, let F^d be an ERDF d-formula, and let F be an ERDF formula. Let $n \geq n_O$.

1. The problem of establishing whether O has a $\#n$-stable model is NP-complete w.r.t. the size of O.
2. The problems of establishing whether:
   (i) $O \models^{st_{\#n}} G'$, (ii) $O \models^{st_{\#n}} F^d$
   are co-NP-complete w.r.t. the size of O,
3. Let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) let a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \geq n_O$. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is co-NP-complete w.r.t. the size of O. □

Based on Corollary 1, Theorem 1, and Proposition 9, we provide complexity results regarding the (original) stable model semantics for bounded, objective ERDF ontologies:

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

1. The problem of establishing whether O has a stable model is NP-complete w.r.t. the size of 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. the size of O. □

7.2 Unbounded ontologies

By allowing variables in the body of objective and simple rules, complexity of the reasoning tasks increases to the second level of the polynomial hierarchy. We will show this by a reduction from the *2-QBF$_\forall$-problem*. A *2-universal quantified boolean formula* (*2-QBF$_\forall$*) is a formula of the form:

$$\mathcal{F} = \forall ?x_1 ... \forall ?x_k \exists ?y_1 ... \exists ?y_m \; c_1 \wedge ..... \wedge \; c_l, \text{where}$$

each c_i is a clause of three literals from the variables $?x_1, ..., ?x_k, ?y_1, ..., ?y_m$[17]. Deciding if $\mathcal{F}$ is valid (called *2-QBF$_\forall$-problem*) is a Π_2^P =co-NP$^{\text{NP}}$-complete problem [66, 58].

Let $\mathcal{F}$ be a 2-universal quantified boolean formula. We will construct a simple ERDF ontology, denoted by $O_\mathcal{F} = \langle G_\mathcal{F}, P_\mathcal{F} \rangle$, such that $\mathcal{F}$ is invalid iff $O_\mathcal{F}$ has a $\#n$-stable model, for $n \in I\!N$. Thus, based on this reduction, we will show that the satisfiability problem of a simple ERDF ontology $O = \langle G, P \rangle$, under the $\#n$-stable model semantics is Σ_2^P-hard w.r.t. the size of O.

Let $X = \{?x_1, ..., ?x_k\}$ and let $Y = \{?y_1, ..., ?y_k\}$. Additionally, let $s \in \mathcal{URI}$ and let $x_i \in \mathcal{URI}$, for each $?x_i \in X$. Further, by $\mathtt{t}$, we will denote the truth value *true* and by $\mathtt{f}$, we will denote the truth value *false*.

Let $G_\mathcal{F} = \{p(x_i, x_i) \mid i = 1, ..., k\} \cup \{rdf{:}type(IsTrue, erdf{:}TotalClass)\}$. The triples of the form $p(x_i, x_i)$ are included in the graph in order to guarantee that we have in the vocabulary at least one URI for each universally quantified variable, which together with the declaration that $IsTrue$ is a total class will generate all possible assignments of boolean truth-values to those variables. The rules and constraints in

[17] The *not* operator is indicated by $\sim$.

the following program $P_{\mathcal{F}}$ will check if this is an invalid assignment. Let $P_{\mathcal{F}}$ be the ERDF program, containing the following rules (and constraints):

For each clause c_i, we define rules which state the truth assignments to the variables from Y that make the clause true given the truth assignment of the variables from X in c_i. This is best illustrated by examples, consider that $c_i = ?x_j \vee \sim ?x_{j'} \vee ?y_{j''}$. Then, we add to $P_{\mathcal{F}}$ the rules.

$$c_i(s, \mathtt{f}) \leftarrow rdf{:}type(x_j, IsTrue). \qquad c_i(s, \mathtt{t}) \leftarrow rdf{:}type(x_j, IsTrue).$$
$$c_i(s, \mathtt{f}) \leftarrow \neg rdf{:}type(x_{j'}, IsTrue). \qquad c_i(s, \mathtt{t}) \leftarrow \neg rdf{:}type(x_{j'}, IsTrue).$$
$$c_i(s, \mathtt{t}) \leftarrow \neg rdf{:}type(x_j, IsTrue),\ rdf{:}type(x_{j'}, IsTrue).$$

Informally, the first rule states that if $?x_j$ is assigned the truth value *true* and $?y_{j''}$ is assigned the truth value *false* then the clause c_i is satisfied. Similarly, the last rule states that if $?x_j$ is assigned the truth value *false* , $?x_{j'}$ is assigned the truth value *true*, and $y_{j''}$ assigned the truth value *true* then the clause c_i is satisfied. The symbol s in the conclusion of the rules is used for creating ERDF triples.

As another example, consider that $c_i = ?x_j \vee \sim ?y_{j'} \vee ?y_{j''}$. Then, we add to $P_{\mathcal{F}}$ the rules.

$$c_{i,1}(s_{\mathtt{f},\mathtt{f}}, \mathtt{f}) \leftarrow rdf{:}type(x_j, IsTrue). \qquad c_{i,2}(s_{\mathtt{f},\mathtt{f}}, \mathtt{f}) \leftarrow rdf{:}type(x_j, IsTrue).$$
$$c_{i,1}(s_{\mathtt{f},\mathtt{t}}, \mathtt{f}) \leftarrow rdf{:}type(x_j, IsTrue). \qquad c_{i,2}(s_{\mathtt{f},\mathtt{t}}, \mathtt{t}) \leftarrow rdf{:}type(x_j, IsTrue).$$
$$c_{i,1}(s_{\mathtt{t},\mathtt{f}}, \mathtt{t}) \leftarrow rdf{:}type(x_j, IsTrue). \qquad c_{i,2}(s_{\mathtt{t},\mathtt{f}}, \mathtt{f}) \leftarrow rdf{:}type(x_j, IsTrue).$$
$$c_{i,1}(s_{\mathtt{t},\mathtt{t}}, \mathtt{t}) \leftarrow rdf{:}type(x_j, IsTrue). \qquad c_{i,2}(s_{\mathtt{t},\mathtt{t}}, \mathtt{t}) \leftarrow rdf{:}type(x_j, IsTrue).$$
$$c_{i,1}(s_{\mathtt{f},\mathtt{f}}, \mathtt{f}) \leftarrow \neg rdf{:}type(x_j, IsTrue). \qquad c_{i,2}(s_{\mathtt{f},\mathtt{f}}, \mathtt{f}) \leftarrow \neg rdf{:}type(x_j, IsTrue).$$
$$c_{i,1}(s_{\mathtt{f},\mathtt{t}}, \mathtt{f}) \leftarrow \neg rdf{:}type(x_j, IsTrue). \qquad c_{i,2}(s_{\mathtt{f},\mathtt{t}}, \mathtt{t}) \leftarrow \neg rdf{:}type(x_j, IsTrue).$$
$$c_{i,1}(s_{\mathtt{t},\mathtt{t}}, \mathtt{t}) \leftarrow \neg rdf{:}type(x_j, IsTrue). \qquad c_{i,2}(s_{\mathtt{t},\mathtt{t}}, \mathtt{t}) \leftarrow \neg rdf{:}type(x_j, IsTrue).$$

Informally, the first two rules state that if $?x_j$ is assigned the truth value *true*, $?y_{j'}$ is assigned the truth value *false*, and $?y_{j''}$ is assigned the truth value *false* then the clause c_i is satisfied.

In particular, $c_{i,1}(s_{y,y'}, y'')$ expresses that clause c_i is true if the *first* variable in Y appearing in the clause c_i takes the value y'' and the second variable in Y appearing in clause c_i takes the value y'. The value y is the same as the value y''. Similarly, $c_{i,2}(s_{y,y'}, y'')$ expresses that the clause c_i is true if the *second* variable in Y appearing in clause c_i takes the value y'' and the first variable in Y appearing in clause c_i takes the value y. The value y' is the same as the value y''. The reader will notice that at each raw of rules the subject of the conclusion of the rules is the same. This is because based on this common subject the two variables in Y are connected together.

As another example, consider that $c_i = ?y_j \vee \sim ?y_{j'} \vee ?y_{j''}$. Then, we add to $P_{\mathcal{F}}$ the rules.

$$c_{i,1}(s_{\mathtt{t},\mathtt{f},\mathtt{f}}, \mathtt{t}) \leftarrow true. \qquad c_{i,2}(s_{\mathtt{t},\mathtt{f},\mathtt{f}}, \mathtt{f}) \leftarrow true. \qquad c_{i,3}(s_{\mathtt{t},\mathtt{f},\mathtt{f}}, \mathtt{f}) \leftarrow true.$$
$$c_{i,1}(s_{\mathtt{t},\mathtt{t},\mathtt{f}}, \mathtt{t}) \leftarrow true. \qquad c_{i,2}(s_{\mathtt{t},\mathtt{t},\mathtt{f}}, \mathtt{t}) \leftarrow true. \qquad c_{i,3}(s_{\mathtt{t},\mathtt{t},\mathtt{f}}, \mathtt{f}) \leftarrow true.$$
$$c_{i,1}(s_{\mathtt{t},\mathtt{f},\mathtt{t}}, \mathtt{t}) \leftarrow true. \qquad c_{i,2}(s_{\mathtt{t},\mathtt{f},\mathtt{t}}, \mathtt{f}) \leftarrow true. \qquad c_{i,3}(s_{\mathtt{t},\mathtt{f},\mathtt{t}}, \mathtt{t}) \leftarrow true.$$
$$c_{i,1}(s_{\mathtt{t},\mathtt{t},\mathtt{t}}, \mathtt{t}) \leftarrow true. \qquad c_{i,2}(s_{\mathtt{t},\mathtt{t},\mathtt{t}}, \mathtt{t}) \leftarrow true. \qquad c_{i,3}(s_{\mathtt{t},\mathtt{t},\mathtt{t}}, \mathtt{t}) \leftarrow true.$$
$$c_{i,1}(s_{\mathtt{f},\mathtt{f},\mathtt{t}}, \mathtt{f}) \leftarrow true. \qquad c_{i,2}(s_{\mathtt{f},\mathtt{f},\mathtt{t}}, \mathtt{f}) \leftarrow true. \qquad c_{i,3}(s_{\mathtt{f},\mathtt{f},\mathtt{t}}, \mathtt{t}) \leftarrow true.$$
$$c_{i,1}(s_{\mathtt{f},\mathtt{f},\mathtt{f}}, \mathtt{f}) \leftarrow true. \qquad c_{i,2}(s_{\mathtt{f},\mathtt{f},\mathtt{f}}, \mathtt{f}) \leftarrow true. \qquad c_{i,3}(s_{\mathtt{f},\mathtt{f},\mathtt{f}}, \mathtt{f}) \leftarrow true.$$
$$c_{i,1}(s_{\mathtt{f},\mathtt{t},\mathtt{t}}, \mathtt{f}) \leftarrow true. \qquad c_{i,2}(s_{\mathtt{f},\mathtt{t},\mathtt{t}}, \mathtt{t}) \leftarrow true. \qquad c_{i,3}(s_{\mathtt{f},\mathtt{t},\mathtt{t}}, \mathtt{f}) \leftarrow true.$$

Informally, the first three rules state that if $?y_j$ is assigned the truth value *true*, $?y_{j'}$ is assigned the truth value *false*, and $?y_{j''}$ is assigned the truth value *false* then the clause c_i is satisfied.

As final example, consider that $c_i = ?x_j \vee \sim ?x_{j'} \vee ?x_{j''}$. Then, we add to $P_{\mathcal{F}}$ the rules.

$$c_i(s,s) \leftarrow rdf{:}type(x_j, IsTrue). \qquad c_i(s,s) \leftarrow \neg rdf{:}type(x_{j'}, IsTrue).$$
$$c_i(s,s) \leftarrow rdf{:}type(x_{j''}, IsTrue).$$

Informally, the first rule states that if $?x_j$ is assigned the truth value *true* then the clause c_i is satisfied.

Now, we add a rule to $P_{\mathcal{F}}$ whose body corresponds to evaluating the formula[18]: $\exists ?y_1 ... \exists ?y_m \; c_1 \wedge ... \wedge c_l$, based on the truth assignment of $?x_1, ..., ?x_n$:

$$w(s,s) \leftarrow c'_1, ..., c'_l., \text{ where}$$

If c_i has the variables $?x_j$, $?x_{j'}$, and $?y_{j''}$ then $c'_i = c_i(s, ?y_{j''})$. If c_i has the variables $?x_j$, $?y_{j'}$, and $?y_{j''}$ then $c'_i = c_{i,1}(?x, ?y_{j'}), c_{i,2}(?x, ?y_{j''})$. If c_i has the variables $?y_j$, $?y_{j'}$, and $?y_{j''}$ then $c'_i = c_{i,1}(?x, ?y_j), c_{i,2}(?x, ?y_{j'}), c_{i,3}(?x, ?y_{j''})$. If c_i has the variables $?x_j$, $?x_{j'}$, and $?x_{j''}$ then $c'_i = c_i(s,s)$.

Finally, we add the constraint:

$$false \leftarrow w(s,s).$$

Note that variables occur only in the condition part of the rule for $w(s,s)$ being the remaining rules in $P_{\mathcal{F}}$ objective bounded ones.

According to the previous constraint, we will keep models where $w(s,s)$ does not hold, i.e. finding an assignment to variables $?x_1, ..., ?x_n$ that makes $\exists ?y_1 ... \exists ?y_m \; c_1 \wedge ... \wedge c_l$ false. Thus, the following result:

Proposition 10. Let $\mathcal{F} = \forall ?x_1 ... \forall ?x_k \exists ?y_1 ... \exists ?y_m \; c_1 \wedge ..... \wedge \; c_l$ be a 2-universal quantified boolean formula. $\mathcal{F}$ is invalid iff $O_{\mathcal{F}} = \langle G_{\mathcal{F}}, P_{\mathcal{F}} \rangle$ has a $\#n$-stable model, for $n \in I\!N$. □

Proposition 10 will be used to provide the complexity of the query answering problem on a simple ERDF ontology w.r.t. the $\#n$-stable model semantics.

Proposition 11. Let $O = \langle G, P \rangle$ be a simple ERDF ontology and let F be an ERDF formula. Additionally, let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \in I\!N$.

1. The problem of establishing whether O has a $\#n$-stable model is $\Sigma_2^P = \text{NP}^{\text{NP}}$-complete w.r.t. the size of O.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is $\Pi_2^P = \text{co-NP}^{\text{NP}}$-complete w.r.t. the size of O. □

We conclude our complexity analysis stating results for the $\#n$-stable model semantics with 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*, to the instances of which, the OWA applies.

[18] The variable $?x_i$ is assigned the truth value *true* iff $rdf{:}type(x_i, IsTrue)$ holds.

Proposition 12. Let $O = \langle G, P \rangle$ be an objective ERDF ontology. Let G' be an ERDF graph, let F^d be an ERDF *d*-formula, and let F be an ERDF formula. Let $n \geq n_O$.

1. The problem of establishing whether O has a $\#n$-stable model is NP^{NP}-complete w.r.t. the size of O.
2. The problems of establishing whether:
   (i) $O \models^{st_{\#n}} G'$, (ii) $O \models^{st_{\#n}} F^d$
   are co-NP^{NP}-complete w.r.t. the size of O,
3. Let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) let a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \geq n_O$. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is co-NP^{NP}-complete w.r.t. the size of O. □

Based on Corollary 1, Theorem 1, and Proposition 12, we provide complexity results regarding the (original) stable model semantics for objective ERDF ontologies:

Corollary 3. 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. $max(\{i \in I\!N \mid rdf{:}\_i \in V_{\mathtt{X}}\}) \leq n_O$, where $\mathtt{X} \in \{G', F^d\}$.

1. The problem of establishing whether O has a stable model is NP^{NP}-complete w.r.t. the size of O.
2. The problems of establishing whether:
   (i) $O \models^{st} G'$ and (ii) $O \models^{st} F^d$
   are co-NP^{NP}-complete w.r.t. the size of O. □

Yet, as mentioned in Section 4, satisfiability and entailment on *simple* (and of course, general) ERDF ontologies under the ERDF stable model semantics are undecidable. Additionally, in Section 4, we showed that entailment of a general ERDF formula from an objective ERDF ontology under the ERDF stable models semantics is also undecidable.

8 Computing the #*n*-Stable Models of General ERDF Ontologies

In this section we address the problem of computing the $\#n$-stable models of general ERDF ontologies, since they cannot be computed directly with Answer Set Programming because of syntactic limitations imposed to the body of rules (just conjunctions of literals, i.e. simple bodies). Our approach mimics the theoretical definition of $\#n$-stable models, which suffices to prove complexity results. In a nutshell, our algorithm has three main parts. Algorithm 8.1 (*All-#n-StableModelsGeneral*) is a brute force search algorithm which generates every possible $\#n$-semi-Herbrand interpretation for a given ERDF ontology O and n, checking whether this interpretation is a stable model by calling main Algorithm 8.2.

Algorithm 8.2 (*Is-#n-StableModelGeneral*) tries to construct the sequence of interpretations which justifies that its third argument M is an $\#n$-stable model. First, it checks if M is a $\#n$-Herbrand interpretation of the ERDF ontology and if it is a minimal interpretation. Subsequently, it constructs iteratively the sequence starting from skolemised ERDF graph by adding the consequences of the rules whose applicable conditions remain true, making the rule supported by the model M. The

algorithm terminates after the fixpoint been reached, returning TRUE whenever the sequence constructs the initial input model M. Algorithm 8.2 needs to call Algorithm 8.3 (*MiddleNotSatisfies*) which checks if a formula is true in an interval of interpretations.

In particular, Algorithm 6.1 *All-#n-StableModelsGeneral*(O, n) takes as input a general ERDF ontology O and an $n \geq n_O$ and computes the set of all $\#n$-stable models of O. This algorithm calls the Algorithm 6.2 *Is-#n-StableModelGeneral*(O, n, M) that takes as input a general ERDF ontology O, an $n \geq n_O$, and a $\#n$-semi-Herbrand interpretation M of O and returns TRUE, if M is a $\#n$-stable model of O, and FALSE, otherwise.

Algorithm 8.1 *All-#n-StableModelsGeneral*(O, n)

Input: (i) a general ERDF ontology O and (ii) an $n \geq n_O$
Output: the set of all $\#n$-stable models of O

(1) S={};
(2) For all $\#n$-semi-Herbrand interpretations M of O do
(3) If *Is-#n-StableModelGeneral*(O, n, M)=TRUE then $S = S \cup \{M\}$;
(4) Endfor
(5) *return*(S);

Algorithm *Is-#n-StableModelGeneral*(O, n, M) calls Algorithm 6.3 *MiddleNotSatisfies*(O, n, I, M, F) which takes as input (i) a general ERDF ontology O, (ii) an $n \geq n_O$, (iii) two ERDF $\#n$-interpretations I, M of O, and (iv) an ERDF formula F and returns TRUE, if it exists $J \in [I, M]_O^{\#n}$ and $J \not\models F$, and FALSE, otherwise. Thus, *MiddleNotSatisfies*(O, n, I, M, F)=FALSE iff $J \models F$, $\forall J \in [I, M]_O^{\#n}$.

Algorithm *Is-#n-StableModelGeneral*(O, n, M), in line (2), checks if M is a $\#n$-Herbrand interpretation of O, and if it is not then it returns FALSE. In line (3), it computes $N_0 = T^{\uparrow\omega}_{[\Pi_O^{H\#n}]^N}(T_{\Pi_G}(\emptyset))$.[19] Thus, $ELP^{-1}(N_0) \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \models sk(G)\})$. This corresponds to item 1 of Definition 9 (ERDF $\#n$-stable model). In line (4), it checks if $M < ELP^{-1}(N_0)$, and if this is the case then it returns FALSE. Then, in line (6), it enters into a loop for $\alpha \in \{0, 1, ...\}$, where in lines (9-10) it computes $S = N_\alpha \cup \{L_{Concl(r)} \mid r \in [P]_{V_O^{\#n}}$ and $J \models Cond(r), \forall J \in [ELP^{-1}(N_\alpha), M]_O^{\#n}\}$ and in line (12) it computes $N_{\alpha+1} = T^{\uparrow\omega}_{[\Pi_O^{H\#n}]^N}(S)$, until $ELP^{-1}(N_\alpha) = ELP^{-1}(N_{\alpha+1})$ or $ELP^{-1}(N_\alpha) > M$. Thus, $ELP^{-1}(N_\alpha) \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \geq ELP^{-1}(N_{\alpha-1})$ and $\forall\ r \in [P]_{V_O^{\#n}}$, if $J \models Cond(r)$, $\forall J \in [ELP^{-1}(N_{\alpha-1}), M]_O^{\#n}$, then $I \models Concl(r)\})$. This corresponds to item 2 of Definition 9 (ERDF $\#n$-stable model). If there is $\alpha \in \{0, 1, ...\}$ such that $ELP^{-1}(N_\alpha) = ELP^{-1}(N_{\alpha+1}) = M$ then *Is-#n-StableModelGeneral*(O, n, M) returns TRUE else it returns FALSE. The user can notice that $ELP(.)$ and $ELP^{-1}(.)$ are used in the algorithm for translating a $\#n$-semi-Herband Interpretation M to an equivalent set S of consistent ELP literals $H(s, p, o)$ and $\neg H(s, p, o)$, where $s, p, o \in V_O^{\#n}$ and vice versa.

[19] $T_P^{\uparrow\omega}(S)$ denotes that the operator T_P is applied from the set S until closure [47].

Algorithm 8.2 *Is-#n-StableModelGeneral*(O, n, M)

Input: (i) a general ERDF ontology $O = \langle G, P\rangle$, (ii) an $n \geq n_O$, and
(iii) a #n-semi-Herbrand interpretation M of O
Output: TRUE, if M is a #n-stable model of O, and FALSE, otherwise

(1) Let $N = ELP(M)$;
(2) If $T_{[\Pi_O^{H\#n}]^N}(N) \neq N$ then return(FALSE);
/* In this case, M is not a #n-Herbrand interpretation of O */
(3) Let $N_0 = T^{\uparrow\omega}_{[\Pi_O^{H\#n}]^N}(T_{\Pi_G}(\emptyset))$;
/* Note that $ELP^{-1}(N_0)$ is a #n-Herbrand interpretation of O */
(4) If $N \subset N_0$ then *return*(FALSE);
(5) Let $\alpha = 0$;
(6) While $N_\alpha \subseteq N$ do
{
(7) $S = N_\alpha$;
(8) $I = ELP^{-1}(N_\alpha)$;
(9) For each $r \in [P]_{V_O^{\#n}}$ do
{
(10) If *MiddleNotSatisfies*$(O, n, I, M, Cond(r))$=FALSE then $S = S \cup \{L_{Concl(r)}\}$;
}
(11) $\alpha = \alpha + 1$;
(12) Let $N_\alpha = T^{\uparrow\omega}_{[\Pi_O^{H\#n}]^N}(S)$;
/* Note that $ELP^{-1}(N_\alpha)$ is a #n-Herbrand interpretation of O */
(13) If $N_\alpha = N_{\alpha-1}$ then
(14) If $N_\alpha = N$ then *return*(TRUE);
(15) else *return*(FALSE);
}
(16) *return*(FALSE);

Algorithm 8.3 *MiddleNotSatisfies*(O, n, I, M, F)

Input: (i) a general ERDF ontology O, (ii) an $n \geq n_O$,
(iii) I,M are ERDF #n-interpretations of O, and (iv) F is an ERDF formula with $FVar = \{\}$
Output: TRUE, if it exists $J \in [I, M]_O^{\#n}$ and $J \not\models F$, and FALSE, otherwise

(1) If a #n-semi-Herbrand interptetation J of O exists s.t.
(2) (i) $T_{[\Pi_O^{H\#n}]^N}(N) \subseteq N$, where $N = ELP(J)$,
/* i.e. J is a #n-Herbrand interpretation of O */
(3) (ii) $I \leq J \leq M$, and
(4) (iii) *Satisfies*(J, F)=FALSE
(5) then *return*(TRUE);
(6) else *return*(FALSE);

The following proposition proves formally correctness of Algorithm *Is-#n-StableModelGeneral*(O, n, M).

Theorem 3. Let O be a general ERDF ontology and let $n \geq n_O$. Let M be a $\#n$-semi-Herbrand-interpretation of O. It is the case that: $M \in \mathcal{M}^{st_{\#n}}(O)$ iff *Is-#n-StableModelGeneral*(O, n, M)=TRUE. □

Example 8. Consider the ERDF ontology $O = \langle G, P\rangle$ of Example 4 and the single #1-stable model M of O, presented in Example 6. Assume now that we call *Is-#n-StableModelGeneral*$(O, 1, M)$. Then, $H(Peter, rdf{:}type, HappyParent) \in N_1$. This is due to the single rule of P. Since $N_2 = N_1 = ELP(M)$, *Is-#n-StableModelGeneral*$(O, 1, M)$ returns TRUE. □

Example 9. Consider the ERDF ontology $O = \langle G, P\rangle$ of Example 1 and a #1-stable model $M_1 \in \mathcal{M}_1$ of the ERDF ontology O, presented in Example 7. Assume now that we call *Is-#n-StableModelGeneral*$(O, 1, M_1)$. Then, $\neg H(Anne, rdf{:}type, Adult) \in N_1$. This is due to rule (3) of P and the rule:
$\neg H(?z, rdf{:}type, ?x) \leftarrow H(?x, rdf{:}type, erdf{:}TotalClass), \sim H(?z, rdf{:}type, ?x)$
of $\Pi_O^{H_{\#n}}$. Additionally, $\neg H(Anne, rdf{:}type, Child) \in N_1$. This is due to rule (4) of P. Further, $H(Retsina, rdf{:}type, SelectedWine) \in N_1$. This is due to rule (2) of P. Now note that $N_2 = N_1 \cup \{H(Anne, serveSoftDrink, Coca\text{-}Cola)\}$. This is due to rule (5) of P. Since $N_3 = N_2 = ELP(M_1)$, *Is-#n-StableModelGeneral*$(O, 1, M_1)$ returns TRUE. □

9 Complexity Results for General ERDF Ontologies

In this section, we provide complexity results for the ERDF $\#n$-stable model semantics on ERDF ontologies without quantifiers and on general ERDF ontologies. We also consider the case of bounded ERDF ontologies without quantifiers and bounded ERDF ontologies.

A *2-quantified boolean formula* (*2-QBF*) is a formula of the form:

$$F = \exists ?x_1 ... \exists ?x_k \forall ?y_1 ... \forall ?y_m R(?x_1, ..., ?x_k, ?y_1, ..., ?y_m), \text{where}$$

$R(?x_1, ..., ?x_k, ?y_1, ..., ?y_m)$ is an unquantified boolean expression[20], in disjunctive normal form, over the variables $?x_1, ..., ?x_k$, $?y_1, ..., ?y_m$. Deciding if F is valid (called *2-QBF-problem*) is a $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$-complete problem [66, 58].

Let F be a 2-quantified boolean formula. We will construct a bounded, ERDF ontology without quantifiers in the body of the rules, denoted by $O_F = \langle G_F, P_F\rangle$, such that F is valid iff O_F has a $\#n$-stable model, for $n \in \mathbb{N}$. Thus, based on this reduction, we will show that the satisfiability problem of a bounded, ERDF ontology $O = \langle G, P\rangle$ without quantifiers, under the $\#n$-stable model semantics is Σ_2^P-hard.

Let $F = \exists ?x_1 ... \exists ?x_k \forall ?y_1 ... \forall ?y_m R(?x_1, ..., ?x_k, ?y_1, ..., ?y_m)$ be a 2-quantified boolean formula. Let $s \in \mathcal{URI}$. We denote by f_R the ERDF formula $f_R = R(p_1(s, s), ..., p_k(s, s), q_1(s, s), ..., q_m(s, s))$.

Let $G_F = \{\}$ and let P_F be the ERDF program, containing the following rules (and constraints):

[20] The *not* operator in R is indicated by $\sim$.

$$
\begin{array}{lll}
\neg p_i(s,s) & \leftarrow & \sim p_i(s,s). \\
p_i(s,s) & \leftarrow & \sim \neg p_i(s,s), \text{ for } i = 1, ..., k. \\
 & & \\
w(s,s) & \leftarrow & f_R. \\
q_i(s,s) & \leftarrow & f_R, \text{ for } i = 1, ..., m. \\
 & & \\
\mathit{false} & \leftarrow & \sim w(s,s).
\end{array}
$$

Proposition 13. Let $F = \exists ?x_1 ... \exists ?x_k \forall ?y_1 ... \forall ?y_m R(?x_1, ..., ?x_k, ?y_1, ..., ?y_m)$ be a 2-quantified boolean formula. F is valid iff $O_F = \langle G_F, P_F \rangle$ has a $\#n$-stable model, for $n \in I\!N$. □

The following proposition provides the complexity of the $\#n$-stable model semantics of bounded, ERDF ontologies without quantifiers.

Proposition 14. Let $O = \langle G, P \rangle$ be a bounded, ERDF ontology without quantifiers and let F be an ERDF formula. Additionally, let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \in I\!N$.

1. The problem of establishing whether O has a $\#n$-stable model is Σ_2^P =NP$^{\text{NP}}$-complete w.r.t. the size of O.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is Π_2^P =co-NP$^{\text{NP}}$-complete w.r.t. the size of O. □

The following proposition provides the complexity of the $\#n$-stable model semantics of ERDF ontologies without quantifiers.

Proposition 15. Let $O = \langle G, P \rangle$ be an ERDF ontology without quantifiers and let F be an ERDF formula. Additionally, let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \in I\!N$.

1. The problem of establishing whether O has a $\#n$-stable model is Σ_2^P =NP$^{\text{NP}}$-complete w.r.t. the size of O.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is Π_2^P =co-NP$^{\text{NP}}$-complete w.r.t. the size of O. □

A *fully quantified boolean formula* (*QBF*) is a formula of the form:

$$\Phi = \exists ?x_1 \forall ?x_2 ... Q_k ?x_k R(?x_1, ..., ?x_k), \text{where}$$

$R(?x_1, ..., ?x_k)$ is an unquantified boolean expression[21], in conjunctive normal form, over the variables $?x_1, ..., ?x_k$. Deciding if Φ is valid (called *QBF-problem*) is a PSPACE-complete problem [66, 58].

Let Φ be a fully quantified boolean formula. We will construct a bounded ERDF ontology, denoted by $O_\Phi = \langle G_\Phi, P_\Phi \rangle$, such that Φ is valid iff O_Φ has a $\#n$-stable model, for $n \in I\!N$. Thus, based on this reduction, we will show that the satisfiability problem of a bounded ERDF ontology $O = \langle G, P \rangle$, under the $\#n$-stable model semantics, is PSPACE-hard.

[21] The *not* operator in R is indicated by $\sim$.

Let $\Phi = \exists ?x_1 \forall ?x_2 ... Q_k ?x_k R(?x_1, ..., ?x_k)$ be a fully quantified boolean formula. Let $s \in \mathcal{URI}$. We denote by ϕ_R the ERDF formula $\phi_R = \exists ?x_1 \forall ?x_2 ... Q_k ?x_k R(p_1(?x_1, s), ..., p_k(?x_k, s))$.

Let $G_\Phi = \{\}$ and let P_Φ be the ERDF program, containing the following rules (and constraints):

$$
\begin{array}{lll}
p_i(s,s) & \leftarrow & \textit{true}, \text{ for } i = 1, ..., k. \\
 & & \\
w(s,s) & \leftarrow & \phi_R. \\
\textit{false} & \leftarrow & \sim w(s,s).
\end{array}
$$

Proposition 16. Let $\Phi = \exists ?x_1 \forall ?x_2 ... Q_k ?x_k R(?x_1, ..., ?x_k)$ be a fully quantified boolean formula. Φ is valid iff $O_\Phi = \langle G_\Phi, P_\Phi \rangle$ has a $\#n$-stable model, for $n \in \mathbb{N}$. □

The following proposition provides the complexity of the $\#n$-stable model semantics of bounded ERDF ontologies. The PSPACE-completeness result is due to the possible quantifiers in the condition part of the rules of a bounded ERDF ontology.

Proposition 17. Let $O = \langle G, P \rangle$ be a bounded ERDF ontology and let F be an ERDF formula. Additionally, let v be (i) "yes", 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 a $\#n$-stable model is PSPACE-complete w.r.t. the size of O.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is PSPACE-complete w.r.t. the size of O. □

The following proposition provides the complexity of the $\#n$-stable model semantics of general ERDF ontologies.

Proposition 18. Let $O = \langle G, P \rangle$ be a general ERDF ontology and let F be an ERDF formula. Additionally, let v be (i) "yes", 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 a $\#n$-stable model is PSPACE-complete w.r.t. the size of O.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is PSPACE-complete w.r.t. the size of O. □

10 Combined Complexity for all Kinds of ERDF Ontologies

Up to now we have considered the complexity of query answering w.r.t. the size of the ERDF ontology. Below, we consider the complexity of query answering w.r.t. both the size of the ERDF ontology and the size of the query formula (i.e. the combined complexity of query answering). We investigate both the case that the query formula does not have quantifiers and the case of a general ERDF query formula.

Proposition 19. Let $O = \langle G, P \rangle$ be a general ERDF ontology, let F' be an ERDF formula without quantifiers, and let F be an ERDF formula. Additionally, let v be (i) "yes", 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 $v \in Ans_O^{st_{\#n}}(F')$ has the same complexity w.r.t. the size of O and F as the complexity w.r.t. the size of O, for all kinds of ERDF ontologies considered in this work.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is PSPACE-complete w.r.t. the size of O and F, for all kinds of ERDF ontologies considered in this work. □

The importance of these new results is that we were able to show that query answering with general ERDF ontologies and general ERDF query formulas under the $\#n$-stable model semantics is no more complex than reasoning with SPARQL over ordinary RDFS graphs. In particular, the combined complexity of SPARQL querying is known to be PSPACE-complete[22] [59]. Thus, we define more complex entailment regimes over more complex ontologies without increasing the complexity of already adopted languages. Additionally, we do RDFS entailment and not only simple entailment with no extra complexity. Further, one can consider alternative scenarios with incomplete information in a more natural fashion via alternative model generation.

11 Related Work

In this Section, we briefly review recent extensions of web ontology languages with rules. In [6], we reviewed and compared ERDF with the following older extensions of web ontology languages with rules [68, 67, 65, 70, 13, 1, 8, 10, 18, 43, 54, 21, 22, 61, 62]. Additionally, at the end of this Section, we provide a comparison of the $\#n$-stable model semantics on general ERDF ontologies with other semantics of logic programs with rules of a richer (than normal programs) syntax.

RDFLog [15] is a recursive, rule-based RDF query language that allows full arbitrary quantifier alternation in the front of the rules and blank nodes in rule heads. However, existential variables should only appear in the head of the rules. RDFLog extends a subset of RDFS [38, 32], called *ρdf* [55]. RDFLog rules should be *range restricted*, that is, (i) if a variable in the head is universal then it should appear in the body of the rule and (ii) if a variable in the head is existential, in the scope of a variable x, then x should also appear in the body of the rule. However, in this work, weak and strong negation are not considered. Thus, closed-world reasoning is not supported. Compared to RDFLog, ERDF extends RDFS. Additionally, through the metaclasses *erdf*:*TotalProperty* and *erdf*:*TotalClass* and the presence of strong and weak negation, it supports open-world and closed-world reasoning on a selective basis for all properties and classes. RDFLog is a very expressive language, with entailment being already EXPTIME-complete for ontologies with blank nodes in data only [24].

Network Graphs [64] is a declarative mechanism to define RDF graphs both extensionally by listing statements and intentionally by using views on other graphs. The proposed mechanism is very powerful and flexible, as it allows one to use almost all of the expressiveness of SPARQL CONSTRUCT [60] queries, including negation and - when used insided Network Graphs - recursive views. However, SPARQL cannot be used to query triples entailed from subclass, subproperty, range, domain, and other relations which can be represented using RDFS [32]. This is because the SPARQL specification defines results of queries based on RDF simple entailment [32]. Additionally, in our theory, the condition of a rule is any ERDF formula over a vocabulary

[22] The combined complexity of SPARQL querying is also PSPACE-complete for graph pattern expressions constructed by using only AND, FILTER, and OPT operators.

V, (thus, involving any of the connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$, $\forall$, and $\exists$) and the head can be a strongly-negated ERDF triple. Combined complexity for Network Graphs evaluation is EXPTIME-complete, while data complexity is polynomial.

Description Logic Rules [40], a subset of SWRL [36], is a rule language for combining Description Logics (DLs) with first-order rules, while the combination remains decidable and can be completely internalized within the DL $\mathcal{SROIQ}$ [35]. However due to the latter fact, the bodies of the rules are conjunctions of atoms and the head is an atom. Additionally, the rules are of a very restricted form, representing the regularity constraints of the role inclusion axioms (RIA) of $\mathcal{SROIQ}$. Complexity results depend on the underlying Description Logic used, but authors are more concerned with tractable fragments.

In [51], an $\mathcal{ALC}^u_{\mathbb{P}}$ knowledge base consists of an $\mathcal{ALC}$ DL and a PBox $\mathbb{P}$ of general rules that share predicates with DL concepts and DL roles. To model open answer set semantics, extended Herbrand structures are used for interpreting DL concepts and DL roles, while open answer sets hold for the general rules. The head of the rules is a DL atom and the bodies of the rules is a conjunction of DL atoms and weakly-negated DL atoms. Rules should satisfy the Datalog safeness condition that each variable of the rule should appear in one of the positive DL atoms in the body of the rules. Additionally, to avoid undecidability, a semantic weak safeness condition is adopted that relies on grounding the variables in the head of the rules with only named individuals. Compared to $\mathcal{ALC}^u_{\mathbb{P}}$, ERDF does not pose a safeness condition. Additionally, in ERDF, the condition of a rule is any ERDF formula over a vocabulary V, (thus, involving any of the connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$, $\forall$, and $\exists$) and the head can be a strongly-negated ERDF triple. Further, ERDF allows the derivation of metalevel statements such as subclass and subproperty relationships, property and class totalness. Authors provide algorithms for the entailment and satisfiability problems which run in 3EXPTIME w.r.t. the size of the $\mathcal{ALC}^u_{\mathbb{P}}$ knowledge base.

In [19], a *hybrid program* is a pair of a description logic knowledge base with a set of hybrid rules. The head of a hybrid rule is a datalog atom and the body is a conjunction of a DL constraint, a set of datalog atoms, and a set of weakly-negated datalog atoms. Hybrid rules should be safe, that is, each variable in the head of the rule, each variable in one of the weakly-negated datalog atoms in the body of the rule, and each free variable in the DL constraint in the body of the rule should be bound in the DL constraint to a constant or to a variable appearing in one of the positive datalog atoms in the body of the rule. The proposed semantics for a hybrid program is defined as a generalization of the well-founded semantics of normal programs [27]. However, in this work, rule-based reasoning is supported only for datalog atoms. In contrast, in ERDF, properties and classes appearing in ERDF graphs can freely appear in the heads and bodies of the rules. No complexity results are provided.

$\mathcal{DL}$+*log* [63] is the general framework for the integration of Description Logics and disjunctive datalog. In particular, a $\mathcal{DL}$+*log* KB is a pair a description logic knowledge base and a disjunctive logic program. The head of a rule is a disjunction of DL atoms or datalog atoms and the body of the rule is a conjunction of datalog atoms, DL atoms, and weakly negated datalog atoms. The datalog safeness condition is imposed, that is every variable occurring in a rule must appear in one of the positive atoms in the body of the rule. Additionally, the weak safeness condition is imposed, that is every variable appearing in the head of the rule must appear in one of the positive datalog atoms in the body of the rule. According to the proposed NM-semantics for $\mathcal{DL}$+*log* KBs, DL predicates are still interpreted under the clas-

sical open-world assumption, while the datalog predicates are interpreted under the closed-world-assumption. Compared to $\mathcal{DL}$+*log*, ERDF does not impose a safety condition. Additionally, through the metaclasses *erdf*:*TotalProperty* and *erdf*:*TotalClass* and the presence of strong and weak negation, it supports open-world and closed-world reasoning on a selective basis for all properties and classes. Data complexity of satisfiability problems ranges from PTIME-complete to Σ_2^2-complete for DL-lite description logics.

The $\mathcal{DL}$*clog* framework [69] extends the $\mathcal{DL}$+*log* [63] framework by allowing weakly negated DL atoms in the bodies of the disjunctive datalog rules. A DL-safeness condition is applied to the rules of $\mathcal{DL}$*clog* KBs, where every variable of a rule must appear in at least one of the positive datalog atoms in the body of the rule. The Nonmonotonic Circumscriptive (NMC) Semantics for $\mathcal{DL}$*clog* KBs is based on a hybrid, modular semantics integrating classical semantics, circumscription [50], and stable model semantics [31]. In particular, in NMC-semantics, DL atoms in rules are evaluated under the circumscriptive models of the DL ontology in the sense of the McCarthy's parallel circumscription [50]. Complexity of satisfiability is NEXPTIME$^{\text{NP}}$-complete.

In [48], a combination of Description Logics and disjunctive logic programs is proposed, which guarantees decidability of reasoning without assuming syntactic restrictions. In particular, a *disjunctive dl-program* is a pair of a description logic knowledge base and a disjunctive logic program. The head of a rule is a disjunction of atoms and the body of the rule is a conjunction of atoms and weakly-negated atoms, where an atom can be a DL atom or a datalog atom. The proposed answer set semantics for disjunctive dl-programs faithfully extends the answer set semantics on disjunctive logic programs [31], when the DL component is empty, and the first-order semantics of description logic knowledge bases, when no rules are present. Complexity for satisfiability is NEXPTIME$^{\text{NP}}$-complete and combined complexity of entailment is co-NEXPTIME$^{\text{NP}}$-complete.

In [52, 53], a framework of *hybrid MKNF knowledge bases* that faithfully integrates Description Logics and disjunctive logic programming is presented, using the logic of Minimal Knowledge and Negation as Failure (MKNF) [44]. A hybrid MKNF knowledge base is a pair of a description logic knowledge base and a set of MKNF rules of the form: $\mathbf{K}\ H_1 \vee ... \vee \mathbf{K}\ H_n \leftarrow \mathbf{K}\ B_1^+, ..., \mathbf{K}\ B_m^+$, **not** $B_1^-, ...,$ **not** B_k^-, where H_i, B_i^+, B_i^- are DL or non-DL atoms and $\mathbf{K}$ and **not** are modal operators. The semantics of a hybrid MKNF knowledge base $\mathcal{K}$ are defined by translating $\mathcal{K}$ into a first-order MKNF formula. To achieve decidability a DL-safety condition is imposed, where every variable of an MKNF rule should appear in at least one non-DL $\mathbf{K}B$ atom in the body of the rule. The proposed semantics for hybrid MKNF knowledge bases faithfully extends the stable model semantics on disjunctive logic programs [31], when the DL component is empty, and the first-order semantics of description logic knowledge bases, when no rules are present. Combined complexity of entailment is at least NEXPTIME$^{\text{NP}}$-complete, depending on the underlying description logic.

In [39], a *coherent well-founded model* for hybrid MKNF knowledge bases [52] is proposed, where disjunction in the head of the MKNF rules is not considered. The same DL-safety condition on the MKNF rules, as that imposed in [52], is also imposed here. This semantics soundly approximates the semantics of [52] and is in a strictly lower complexity class, making a polynomial number of queries to the underlying description logic reasoner. Additionally, it faithfully extends the well-

founded semantics of normal programs [28], when the DL component is empty, and the first-order semantics of description logic knowledge bases, when no rules are present. Assuming that entailment of ground DL-atoms in DL is decidable with data complexity $\mathcal{C}$, the data complexity of computing the well-founded partition is in $\mathrm{P}^{\mathcal{C}}$.

Compared to [69, 48, 53, 39], in ERDF, the condition of a rule is any ERDF formula over a vocabulary V, (thus, involving any of the connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$, $\forall$, and $\exists$) and the head can be a strongly-negated ERDF triple. Further, ERDF allows the derivation of metalevel statements such as subclass and subproperty relationships, property and class totalness.

In [23], the stable models of first-order formulas are defined. Comparing our work with that of [23], we realized that we achieve different results. For example, consider the ERDF ontology $O =< \{\}, P >$, where $P = \{p(s,s) \leftarrow \sim\sim p(s,s).\}$. Then, there is a unique $\#n$-stable model M of O such that $M \models \sim p(s,s)$ while, according to [23], there are two stable models of P: $\{\}$ and $\{p(s,s)\}$. Additionally, consider the ERDF ontology $O =< \{\}, P >$, where $P = \{q(s,s) \leftarrow p(s,s) \vee \sim p(s,s)., p(s,s) \leftarrow q(s,s).\}$. Then, there is a unique $\#n$-stable model M of O such that $M \models p(s,s) \wedge q(s,s)$, while, according to [23], there are no stable models of P. In [41], first-order loop formulas in the context of first-order reasoning were defined. The same comments with [23] apply also here.

12 Conclusions

In this paper, we elaborated on the computability and complexity issues of the ERDF stable model semantics of ERDF ontologies. Based on the work in [6], decidability under this semantics cannot be achieved, unless ERDF ontologies of restricted syntax are considered. We proposed the *ERDF #n-stable model semantics* of ERDF ontologies (for $n \in I\!N$) and showed that entailment under this semantics extends RDFS entailment. Moreover, we showed that query answering under the ERDF $\#n$-stable model semantics is decidable. Equivalence statements between the ERDF stable and $\#n$-stable model semantics for objective ERDF ontologies were provided. We showed that if O is a simple ERDF ontology then query answering under the ERDF $\#n$-stable model semantics reduces to query answering under the answer set semantics [30]. We provided algorithms that compute the ERDF $\#n$-stable models of simple and general ERDF ontologies. Further, we provided complexity results for the ERDF $\#n$-stable model semantics on objective and simple ERDF ontologies, ERDF ontologies without quantifiers, and general ERDF ontologies. We consider both the case that an ERDF ontology is bounded or is not bounded. In particular, the complexity of query answering under the ERDF $\#n$-stable model semantics (i) on bounded objective and bounded simple ERDF ontologies is co-NP-complete, (ii) on objective and simple ERDF ontologies, as well as on bounded ERDF ontologies without quantifiers and ERDF ontologies without quantifiers, is Π_2^P=co-$\mathrm{NP}^{\mathrm{NP}}$-complete, and (iii) on bounded ERDF ontologies and general ERDF ontologies is PSPACE-complete. All previous results are w.r.t. the size of the ERDF ontology.

Additionally, we provided combined complexity results of query answering according to ERDF $\#n$-stable model semantics. In particular, in the case that the query formula does not have quantifiers, the combined complexity of query answering is the same as the complexity of query answering w.r.t. the size of the ERDF ontology, for all kinds of ERDF ontologies. Further, in the case that the query formula is general, the combined complexity of query answering is PSPACE-complete, for all kinds of

ERDF ontologies. All of these complexity results are summarized in Table 1 in the introductory section.

Finally, we provided complexity results for the (original) ERDF stable model semantics on bounded objective and objective ERDF ontologies. In particular, the complexity of query answering under the ERDF stable model semantics on bounded objective ERDF ontologies is co-NP-complete and on objective ERDF ontologies is Π_2^P-complete, provided that the query formula is an ERDF d-formula.

Note that by allowing general ERDF formulas in the bodies of the rules of an ERDF ontology, the user can specify his/her need in a concise and straightforward way. For example, consider the ERDF ontology $O = \langle G, P \rangle$ of Example 4, where P is defined as follows:

$$rdf{:}type(?x, HappyParent) \leftarrow isParentOf(?x, ?y), \\ \forall ?y\ isParentOf(?x, ?y) \supset (\exists ?z\ isMarriedTo(?y, ?z),\ \exists ?w\ isParentOf(?y, ?w)).$$

An ERDF ontology equivalent to O is the simple ERDF ontology $O' = \langle G, P' \rangle$, where P' is defined as follows:

$$\begin{array}{ll} rdf{:}type(?x, HappyParent) & \leftarrow isParentOf(?x, ?y), {\sim}rdf{:}type(?x, Aux). \\ rdf{:}type(?x, Aux) & \leftarrow isParentOf(?x, ?y), {\sim}rdf{:}type(?y, IsMarried). \\ rdf{:}type(?x, Aux) & \leftarrow isParentOf(?x, ?y), {\sim}rdf{:}type(?y, HasChild). \\ rdf{:}type(?x, IsMarried) & \leftarrow isMarriedTo(?x, ?y). \\ rdf{:}type(?x, HasChild) & \leftarrow isParentOf(?x, ?y). \end{array}$$

Obviously, the desire of the user is expressed more straightforwardly through the ERDF ontology O than the simple ERDF ontology O'. Moreover, consider the ERDF ontology $O_\Phi = \langle G_\Phi, P_\Phi \rangle$, defined above Proposition 16, which contains existential and universal quantifiers. Due to Proposition 16, deciding if O_Φ has a $\#n$-stable model is PSPACE-complete. Therefore, due to Proposition 17.1, there is no equivalent to O_Φ[23], simple ERDF ontology of size polynomial in the size of O_Φ.

In [2, 4], we propose a framework for modular ERDF ontologies, called *modular ERDF framework*, which enables collaborative reasoning over a set of ERDF ontologies, while support for hidden knowledge is also provided. In particular, the *modular ERDF stable model semantics* of modular ERDF ontologies is defined, extending the ERDF $\#n$-stable model semantics. In [4], we provide algorithms that compute the proposed modular ERDF stable model semantics and complexity results. Future work concerns the implementation of our ERDF framework and its application to practical problems, such as planning [45], biomedicine, applications on science and humanities, and industrial applications [46, 14].

Appendix A: Proofs

In this appendix, we provide proofs for the propositions not presented in the main paper. The proof of the rest of the propositions is included in the main text. To reduce the size of the proofs, we have eliminated the namespace from the URIs in $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$.

[23] We say that two ERDF ontologies are *equivalent* if they have the same $\#n$-stable models (for $n \in \mathbb{N}$).

Proposition 2 Let $\mathcal{D}$ be an instance of the unbounded tiling problem. It holds: $\mathcal{D}$ has a solution iff $O_\mathcal{D} \not\models^{st} F_\mathcal{D}$.
Proof:
$\Rightarrow$) Let τ be a solution to $\mathcal{D}$. Since $\mathbb{N} \times \mathbb{N}$ is denumerable, there exists a bijective function $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Let I_0 be the minimal Herbrand interpretation of $O_\mathcal{D}$ such that:

1. $CT_{I_0}(Tile) = \{rdf{:}\_i \mid i \in \mathbb{N}\}$ and $CF_{I_0}(Tile) = \emptyset$.
2. $PT_{I_0}(HConstraint) = H$ and $PF_{I_0}(HConstraint) = \emptyset$.
3. $PT_{I_0}(VConstraint) = V$ and $PF_{I_0}(VConstraint) = \emptyset$.
4. $CT_{I_0}(erdf{:}TotalProperty) = \{ofType, right, above\}$ and $CF_{I_0}(erdf{:}TotalProperty) = \emptyset$.
5. $PT_{I_0}(ofType) = \{\langle rdf{:}\_\pi(i,j), \tau(i,j)\rangle \mid i,j \in \mathbb{N}\}$ and
   $PF_{I_0}(ofType) = Res^H_{O_\mathcal{D}} \times Res^H_{O_\mathcal{D}} - PT_{I_0}(ofType)$.
6. $PT_{I_0}(right) = \{\langle rdf{:}\_\pi(i,j), rdf{:}\_\pi(i+1,j)\rangle \mid i,j \in \mathbb{N}\}$ and
   $PF_{I_0}(right) = Res^H_{O_\mathcal{D}} \times Res^H_{O_\mathcal{D}} - PT_{I_0}(right)$.
7. $PT_{I_0}(above) = \{\langle rdf{:}\_\pi(i,j), rdf{:}\_\pi(i,j+1)\rangle \mid i,j \in \mathbb{N}\}$ and
   $PF_{I_0}(above) = Res^H_{O_\mathcal{D}} \times Res^H_{O_\mathcal{D}} - PT_{I_0}(above)$.
8. $CT_{I_0}(HasTileRight) = CT_{I_0}(HasTileAbove) = CT_{I_0}(HasTileType) = \emptyset$.
   $CF_{I_0}(HasTileRight) = CF_{I_0}(HasTileAbove) = CF_{I_0}(HasTileType) = \emptyset$.
9. $PT_{I_0}(id) = \emptyset$ and $PF_{I_0}(id) = \emptyset$.

Let I_1 be the minimal Herbrand interpretation of $O_\mathcal{D}$ which is the same as I_0, except that:

1. $CT_{I_1}(HasTileRight) = CT_{I_1}(HasTileAbove) = CT_{I_1}(HasTileType) = \{rdf{:}\_i \mid i \in \mathbb{N}\}$.
2. $PT_{I_1}(id) = \{\langle x, x\rangle \mid x \in Res^H_{O_\mathcal{D}}\}$.

It is easy to see that I_1 is a stable model of $O_\mathcal{D}$, generated by the sequence $I_0 < I_1$. Additionally, it is the case that $I_1 \not\models F_\mathcal{D}$. 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 $O_\mathcal{D} = \langle G_\mathcal{D}, P_\mathcal{D}\rangle$ such that $I \not\models F_\mathcal{D}$. Obviously, $CT_I(Tile) = \{rdf{:}\_i \mid i \in \mathbb{N}\}$. Since $O_\mathcal{D} \not\models^{st} F_\mathcal{D}$, it is the case 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 $i,j,k \in \mathbb{N}$, 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. Further, it is the case that each tile is assigned a unique type in $\mathcal{T} = \{T_1, ..., T_n\}$ and this type assignment satisfies the horizontal and vertical adjacency constraints of $\mathcal{D}$. Thus, a solution of $\mathcal{D}$ is $\tau : \mathbb{N} \times \mathbb{N} \to \mathcal{T}$, where $\tau(i,j) = T$ iff $\langle rdf{:}\_\pi(i,j), T\rangle \in PT_I(ofType)$. Since π is a total function and, for all $k \in \mathbb{N}$, tile $rdf{:}\_k$ is assigned a unique type in $\mathcal{T}$, it follows that τ is a total function. □

Proposition 3 Let O be an objective ERDF ontology and let F be a general ERDF formula. The problem $O \models^{st} F$ is undecidable.
Proof: Since the unbounded tiling problem is undecidable [11], it follows directly from Proposition 2 that if O is an objective ERDF ontology, entailment of a *general* ERDF formula F under the ERDF stable model semantics is undecidable. □

Theorem 1 Let O be an objective ERDF ontology and let $n \geq n_O$. Let F^d be an ERDF d-formula s.t. $max(\{i \in I\!N \mid rdf{:}\_i \in V_{F^d}\}) \leq n$. It holds: $O \models^{st} F^d$ iff $O \models^{st_{\#n}} F^d$.

Proof:

$\Leftarrow$) Let $O = \langle G, P\rangle$ and let $O \models^{st} F^d$. We will show that $O \models^{st_{\#n}} F^d$. Specifically, let $I \in \mathcal{M}^{st_{\#n}}(O)$, we will show that $I \models F^d$.

First, we define a mapping $m : Res_O^H \rightarrow Res_O^{H_{\#n}}$ as follows:

$$m(x) = \begin{cases} rdf{:}\_n \text{ if } x \in \{rdf{:}\_i \mid i > n\}, \text{ and} \\ x \qquad \text{otherwise} \end{cases}$$

Note that I is an ERDF $\#n$-interpretation of $V_O^{\#n}$. 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: $IL_J : V_O \cap \mathcal{TL} \rightarrow Res_J$ 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: $J : V_O \rightarrow Res_J$ such that:
  - $J(x) = J_V(x),\ \forall x \in V_O \cap \mathcal{URI}$.
  - $J(x) = x,\ \forall\, x \in V_O \cap \mathcal{PL}$.
  - $J(x) = IL_J(x),\ \forall\, x \in V_O \cap \mathcal{TL}$.
- $Prop_J = \{x \in Res_J \mid m(x) \in Prop_I\}$.
- The mapping $PT_J : Prop_J \rightarrow \mathcal{P}(Res_J \times Res_J)$ is defined as follows:
  $\forall x, y, z \in Res_O^H$, it holds:
  $\langle x, y\rangle \in PT_J(z)$ iff $\langle m(x), m(y)\rangle \in PT_I(m(z))$.

- We define the mapping $PF_J : Prop_J \rightarrow \mathcal{P}(Res_J \times Res_J)$ as follows:
  $\forall x, y, z \in Res_O^H$, it holds:
  $\langle x, y\rangle \in PF_J(z)$ iff $\langle m(x), m(y)\rangle \in PF_I(m(z))$.
- $LV_J = \{x \in Res_J \mid \langle x, J(Literal)\rangle \in PT_J(J(type))\}$.

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

Now, we extend J with the ontological categories:
$Cls_J = \{x \in Res_J \mid \langle x, J(Class)\rangle \in PT_J(J(type))\}$,
$TCls_J = \{x \in Res_J \mid \langle x, J(TotalClass)\rangle \in PT_J(J(type))\}$, and
$TProp_J = \{x \in Res_J \mid \langle x, J(TotalProperty)\rangle \in PT_J(J(type))\}$.
We define the mappings $CT_J, CF_J : Cls_J \rightarrow \mathcal{P}(Res_J)$ 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 show that J is an ERDF interpretation of V_O. Note that $\forall x \in V_O^{\#n}$, it is the case that $m(J(x)) = J(x) = I(x)$.

First, we will show that J satisfies semantic condition 2 of Definition 5 (ERDF interpretation), in a number of steps:

Step 1: Here, we prove that $Res_J = CT_J(J(Resource))$. Obviously, $CT_J(J(Resource)) \subseteq Res_J$. We will show that $Res_J \subseteq CT_J(J(Resource))$. Let $x \in Res_J$. We want to

show that $\langle x, J(Resource)\rangle \in PT_J(J(type))$. It holds: $\langle x, J(Resource)\rangle \in PT_J(J(type))$ iff $\langle m(x), I(Resource)\rangle \in PT_I(I(type))$, which is true, since I is an ERDF $\#n$-interpretation of $V_O^{\#n}$, and thus $m(x) \in Res_I$. Therefore, $x \in CT_J(J(Resource))$. Therefore, $Res_J = CT_J(J(Resource))$.

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

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

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

We will now show that J satisfies semantic condition 3 of Definition 5 (ERDF Interpretation). Let $\langle x, y\rangle \in PT_J(J(domain))$ and $\langle z, w\rangle \in PT_J(x)$. We will show that $z \in CT_J(y)$. It holds $\langle m(x), m(y)\rangle \in PT_I(I(domain))$ and $\langle m(z), m(w)\rangle \in PT_I(m(x))$. Since I is an ERDF $\#n$-interpretation, it holds $\langle m(z), m(y)\rangle \in PT_I(I(type))$. Thus, $\langle z, y\rangle \in PT_J(J(type))$ and $z \in CT_J(y)$.

In a similar manner, we can prove that J also satisfies the rest of the semantic conditions of Definition 5. Thus, J is an ERDF interpretation of V_O.

Moreover, we will show that J is a coherent ERDF interpretation. Assume that this is not the case. Thus, there is $z \in Prop_J$ s.t. $PT_J(z) \cap PF_J(z) \neq \emptyset$. Thus, there are $x, y \in Res_J$ s.t. $\langle x, y\rangle \in PT_J(z) \cap PF_J(z)$, for such a z. It follows that $\langle m(x), m(y)\rangle \in PT_I(m(z))$ and $\langle m(x), m(y)\rangle \in PF_I(m(z))$. But this is impossible, since I is a (coherent) ERDF $\#n$-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 $p, s, o \in V_O^{\#n}$. Since $I \models sk(G)$, it holds $I(p) \in Prop_I$. Thus, $\langle I(p), I(Property)\rangle \in PT_I(I(type))$. Note that $m(J(p)) = J(p) = I(p)$, $m(J(Property)) = J(Property) = I(Property)$, and $m(J(type)) = J(type) = I(type)$. Therefore, it follows that $\langle J(p), J(Property)\rangle \in PT_J(J(type))$ and thus, $J(p) \in Prop_J$. It holds: $\langle J(s), J(o)\rangle \in PT_J(J(p))$ iff $\langle m(J(s)), m(J(o))\rangle \in PT_I(m(J(p)))$ iff $\langle I(s), I(o)\rangle \in PT_I(I(p))$. The last statement is true since $I \models sk(G)$. Thus, $J \models sk(G)$.

Let $r \in [P]_{V_O}$ such that $J \models Cond(r)$, we will show that $J \models Concl(r)$. First, we define a total function $u' : V_O \to V_O$, as follows:

$$m'(x) = \begin{cases} x & \text{if } x \text{ is a well-typed XML literal} \\ m(x) & \text{otherwise} \end{cases}$$

Let r be of the form:
$p'(s', o') \leftarrow p_1(s_1, o_1), ..., p_k(s_k, o_k), \neg p_{k+1}(s_{k+1}, o_{k+1}), ..., \neg p_n(s_n, o_n)$.
It is the case that $J \models p_i(s_i, o_i)$, for $i = 1, ..., k$ and $J \models \neg p_i(s_i, o_i)$, for $i = k+1, ..., n$. Note that $p', p_i \in V_O^{\#n}$. Additionally, note that s', o', s_i, o_i are either terms in $V_O^{\#n}$ or the instantiation of variables. Therefore, $I \models p_i(m'(s_i), m'(o_i))$, for $i = 1, ..., k$ and $I \models \neg p_i(m'(s_i), m'(o_i))$, for $i = k+1, ..., n$. This implies that $\langle I(m'(s')), I(m'(o'))\rangle \in$

$PT_I(m(p'))$. Note that $I(m'(s')) = m(J(s))$, $I(m'(o')) = m(J(o))$, and $J(p') = p'$. Therefore, $\langle m(J(s')), m(J(o'))\rangle \in PT_I(m(J(p')))$. Thus, $\langle J(s'), J(o')\rangle \in PT_J(J(p'))$. It follows from this that $J \models p'(s', o')$. Similarly, if $Concl(r) = \neg p'(s', o')$, we can show that $J \models \neg p'(s', o')$.

Let $K_0 = minimal(\{N \in \mathcal{I}^H(O) \mid N \leq J \text{ and } N \models sk(G)\})$. For $\alpha \in \{1, ...\}$, let $K_\alpha = minimal(\{N \in \mathcal{I}^H(O) \mid N \geq K_{\alpha-1}, N \leq J \text{ and } \forall\, r \in [P]_{V_O}, \text{ if } N \models Cond(r) \text{ then } N \models Concl(r)\})$. Note that K_0 and K_α are well-defined and that it exists λ s.t. $K_\lambda = K_{\lambda+1}$. Let $K = K_\lambda$. Obviously, $K \leq J$. Since O is an objective ERDF ontology, it follows that $K_0 \in minimal(\{N \in \mathcal{I}^H(O) \mid N \models sk(G)\})$ and $K_\alpha \in minimal(\{I \in \mathcal{I}^H(O) \mid N \geq I_{\alpha-1}$ and it is the case that: $\forall\, r \in [P]_{V_O}$, if $N \models Cond(r)$, $\forall N \in [I_{\alpha-1}, K]_O^{\#n}$ then $N \models Concl(r)\})$. Thus, $K \in \mathcal{M}^{st}(O)$ and $K \leq J$.

Since $K \in \mathcal{M}^{st}(O)$ and $O \models^{st} F^d$, it follows that $K \models F^d$. Since F^d is an ERDF d-formula, it is the case that

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

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 $u : Var(F_i) \to Res_O^H$ 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, $J, u \models \neg p(s, o)$.

Now, we define a total function $u' : Var(F_i) \to Res_I$ s.t. $u'(x) = m(u(x))$.

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 $max(\{i \in \mathbb{N} \mid rdf{:}\_i \in V_{F^d}\}) \leq n$, it follows that $V_{F_i} \subseteq V_O^{\#n}$. Thus, $p \in V_O^{\#n}$ and $s, o \in V_O^{\#n} \cup Var$.

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

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

Note that $\forall x \in V_{F_i}$, it holds: $I(x) = m(J(x))$ and $[I + u'](x) = m([J + u](x))$. Moreover, $\forall x \in Var(F_i)$, it holds: $[I + u'](x) = m(u(x)) = m([J + u](x))$. Therefore, it holds:

$\langle [I + u'](s), [I + u'](o)\rangle \in PT_I(I(p))$ iff

$\langle m([J + u](s)), m([J + u](o))\rangle \in PT_I(m(J(p)))$ 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 $I \models F_i$. Therefore, $O \models^{st_{\#n}} F$.

$\Rightarrow$) Let $O \models^{st_{\#}} F^d$. We will show that $O \models^{st} F^d$. Let $I \in \mathcal{M}^{st}(O)$. We will show that $I \models F^d$.

Note that I is an ERDF interpretation of V_O. Based on I, we construct a partial interpretation J of $V_O^{\#n}$ as follows:

- $Res_J = Res_O^{H_{\#n}}$.
- $J_V(x) = x$, for all $x \in V_O^{\#n} \cap \mathcal{URI}$.
- We define the mapping: $IL_J : V_O^{\#n} \cap \mathcal{TL} \rightarrow Res_J$ such that:
  $IL_J(x) = x$, 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 define the mapping: $J : V_O^{\#n} \rightarrow Res_J$ such that:
  - $J(x) = J_V(x),\ \forall x \in V_O^{\#n} \cap \mathcal{URI}$.
  - $J(x) = x,\ \forall\, x \in V_O^{\#n} \cap \mathcal{PL}$.
  - $J(x) = IL_J(x),\ \forall\, x \in V_O^{\#n} \cap \mathcal{TL}$.
- $Prop_J = Prop_I \cap Res_J$.
- $\forall x \in Prop_J,\quad PT_J(x) = PT_I(x)$ and $PF_J(x) = PF_I(x)$.
- $LV_J = \{x \in Res_J \mid \langle x, J(Literal)\rangle \in PT_J(J(type))\}$.

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

As in the proof of $\Leftarrow$), we can show that: (i) J is a (coherent) ERDF $\#n$-interpretation of $V_O^{\#n}$, and (ii) $I \models F^d$. □

Corollary 1 Let $O = \langle G, P\rangle$ be an objective ERDF ontology and let $n \geq n_O$. It is the case that: O has a stable model iff O has a $\#n$-stable model.
Proof: Let $F = p(s,o) \wedge \neg p(s,o)$, for $p, s, o \in \mathcal{URI}$. It follows from Theorem 1 that $O \models^{st} F$ iff $O \models^{st_{\#n}} F$. Thus, O has no stable model iff O has no $\#n$-stable model. Therefore, O has a stable model iff O has a $\#n$-stable model. □

Theorem 2 Let O be a simple ERDF ontology and let $n \geq n_O$. Let M be a $\#n$-semi-Herbrand interpretation of O. It is the case that: $M \in \mathcal{M}^{st_{\#n}}(O)$ iff $ELP(M)$ is a consistent answer set of $\Pi_O^{\#n}$.

Proof: First, we give a definition that will be used throughout the proof. Let Π be an ELP and let C be a set of constants. By $[\Pi]_C$, we denote the instantiation of Π w.r.t. the constants appearing in C.

$\Rightarrow$) Let $M \in \mathcal{M}^{st_{\#n}}(O)$. We will show that $N = ELP(M)$ is a consistent answer set of $\Pi_O^{\#n}$. Since $V_O^{\#n}$ is finite and $M \in \mathcal{M}^{st_{\#n}}(O)$, it follows that there is a sequence of $\#n$-Herbrand interpretations $I_0 \leq ... \leq I_{k+1}$ such that $I_k = I_{k+1} = M$ and:

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

2. For successor ordinals α with $0 < \alpha \leq k+1$:
   $I_\alpha \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \geq I_{\alpha-1}$ and it is the case that:
   $\forall\, r \in [P]_{V_O^{\#n}}$, if $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}$, then $I \models Concl(r)\}$.

Now, we define a sequence $N_0 \subseteq .... \subseteq N_{k+1} \subseteq EHB(\Pi_O^{\#n})$, as follows:

$N_0 = T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(T_{\Pi_G}(\emptyset))$.

$N_\alpha = T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(T_{([\Pi_P]_{V_O^{\#n}})^N}(N_{\alpha-1}))$, where $1 \leq \alpha \leq k+1$.

Lemma: It holds $N_\alpha = ELP(I_\alpha)$, for $\alpha = 0, ..., k+1$.
Proof: We will prove the Lemma, by induction.

First, we will show that $N_0 \subseteq ELP(I_0)$. Since $I_0 \models sk(G)$, it follows that $T_{\Pi_G}(\emptyset) \subseteq ELP(I_0)$. As $I_0 \in \mathcal{I}^{H_{\#n}}(O)$, it follows that $ELP(I_0)$ satisfies all rules in $[\Pi_O^{H_{\#n}}]$. Now, as $ELP(I_0) \subseteq N$, it follows that $ELP(I_0)$ satisfies all rules in $[\Pi_O^{H_{\#n}}]^N$. Moreover, as $T_{\Pi_G}(\emptyset) \subseteq ELP(I_0)$, it follows that $T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(T_{\Pi_G}(\emptyset)) \subseteq ELP(I_0)$. Therefore, $N_0 \subseteq ELP(I_0)$.

Let $H(p, type, TotalProperty) \in N_0$, for $p \in V_O^{\#n}$. As $N_0 \subseteq ELP(I_0) \subseteq ELP(M) = N$, it follows that $H(p, type, TotalProperty) \in N$. Therefore, for all $x, y \in V_O^{\#n}$, $[\neg]H(x, p, y) \in N_0$ iff $[\neg]H(x, p, y) \in N$. Similarly, let $H(c, type, TotalClass) \in N_0$, for $c \in V_O^{\#n}$. Then, $H(c, type, TotalClass) \in N$. Therefore, for all $x \in V_O^{\#n}$, $[\neg]H(x, type, c) \in N_0$ iff $[\neg]H(x, type, c) \in N$. Now, from the above and as N_0 satisfies all rules in $[\Pi_O^{H_{\#n}}]^N$, it follows that N_0 satisfies all rules in $[\Pi_O^{H_{\#n}}]$. Therefore, $ELP^{-1}(N_0)$ satisfies all semantic conditions of a (coherent) $\#n$-Herbrand interpretation of O. Thus, $ELP^{-1}(N_0) \in \mathcal{I}^{H_{\#n}}(O)$. Moreover, $ELP^{-1}(N_0) \models sk(G)$. Therefore, $ELP^{-1}(N_0) \in \{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \models sk(G)\}$. Now as $I_0 \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \models sk(G)\})$ and $N_0 \subseteq ELP(I_0)$, it follows that $N_0 = ELP(I_0)$.

Assumption: We assume that $N_{\alpha-1} = ELP(I_{\alpha-1})$, for an $\alpha \leq k$.

We will show that $N_\alpha = ELP(I_\alpha)$. First, we will show that $N_\alpha \subseteq ELP(I_\alpha)$. Due to assumption $N_{\alpha-1} = ELP(I_{\alpha-1})$ and the fact $I_{\alpha-1} \leq I_\alpha$, it follows that $N_{\alpha-1} \subseteq ELP(I_\alpha)$. Based on (i) the assumption $N_{\alpha-1} = ELP(I_{\alpha-1})$ and (ii) the fact $I_\alpha \models Concl(r)$, for all $r \in [P]_{V_O^{\#n}}$ s.t. $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}$, it follows that $T_{([\Pi_P]_{V_O^{\#n}})^N}(N_{\alpha-1}) \subseteq ELP(I_\alpha)$.

As $I_\alpha \in \mathcal{I}^{H_{\#n}}(O)$, it follows that $ELP(I_\alpha)$ satisfies all rules in $[\Pi_O^{H_{\#n}}]$. As $ELP(I_\alpha) \subseteq N$, it follows that $ELP(I_\alpha)$ satisfies all rules in $[\Pi_O^{H_{\#n}}]^N$. Now as $T_{([\Pi_P]_{V_O^{\#n}})^N}(N_{\alpha-1}) \subseteq ELP(I_\alpha)$, it follows that $T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(T_{([\Pi_P]_{V_O^{\#n}})^N}(N_{\alpha-1})) \subseteq ELP(I_\alpha)$. Therefore, $N_\alpha \subseteq ELP(I_\alpha)$.

Let $H(p, type, TotalProperty) \in N_\alpha$, for $p \in V_O^{\#n}$. As $N_\alpha \subseteq ELP(I_\alpha) \subseteq ELP(M) = N$, it follows that $H(p, type, TotalProperty) \in N$. Therefore, for all $x, y \in V_O^{\#n}$, $[\neg]H(x, p, y) \in N_\alpha$ iff $[\neg]H(x, p, y) \in N$. Similarly, let $H(c, type, TotalClass) \in N_\alpha$, for $c \in V_O^{\#n}$. Then, $H(c, type, TotalClass) \in N$. Therefore, for all $x \in V_O^{\#n}$, $[\neg]H(x, type, c) \in N_\alpha$ iff $[\neg]H(x, type, c) \in N$. Now, from the above and as N_α satisfies all rules in $[\Pi_O^{H_{\#n}}]^N$, it follows that N_α satisfies all rules in $[\Pi_O^{H_{\#n}}]$. Therefore, $ELP^{-1}(N_\alpha)$ satisfies all semantic conditions of a (coherent) $\#n$-Herbrand interpretation of O. Thus, $ELP^{-1}(N_\alpha) \in \mathcal{I}^{H_{\#n}}(O)$. Moreover, $ELP^{-1}(N_\alpha) \geq ELP^{-1}(N_{\alpha-1}) =$

$I_{\alpha-1}$. Now, based on the assumption that $N_{\alpha-1} = ELP(I_{\alpha-1}) \subseteq N$ and the fact that $T_{([\Pi_P]_{V_O^{\#n}})^N}(N_{\alpha-1}) \subseteq N_\alpha$, it follows that $ELP^{-1}(N_\alpha) \models Concl(r)$, for all $r \in [P]_{V_O^{\#n}}$ s.t. $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}$. Therefore, $ELP^{-1}(N_\alpha) \in \{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \geq I_{\alpha-1}$ and $I \models Concl(r)$, for all $r \in [P]_{V_O^{\#n}}$ s.t. $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}\}$. Now as $I_\alpha \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \geq I_{\alpha-1}$ and $I \models Concl(r)$, for all $r \in [P]_{V_O^{\#n}}$ s.t. $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}\})$ and $N_\alpha \subseteq ELP(I_\alpha)$, it follows that $N_\alpha = ELP(I_\alpha)$.
End of Lemma

Therefore, $N_k = N_{k+1} = ELP(M)$. Since M is a coherent $\#n$-Herbrand interpretation, it follows that $N = ELP(M)$ is consistent. Moreover, since $[\Pi_O^{\#n}]^N = \Pi_G \cup ([\Pi_P]_{V_O^{\#n}})^N \cup [\Pi_O^{H_{\#n}}]^N$, it follows that $T^{\uparrow\omega}_{[\Pi_O^{\#n}]^N}(\emptyset) = N$. Therefore, $N = ELP(M)$ is a consistent answer set of $\Pi_O^{\#n}$.

$\Leftarrow$) Let $N = ELP(M)$ be a consistent answer set of $\Pi_O^{\#n}$. We will show that $M \in \mathcal{M}^{st_{\#n}}(O)$. Since N is a consistent answer set of $\Pi_O^{\#n}$, it follows that $T^{\uparrow\omega}_{[\Pi_O^{\#n}]^N} = N$. We define a sequence $N_\alpha \subseteq EHB(\Pi_O^{\#n})$, $\alpha \in \{0, 1, ...\}$, as follows:

$N_0 = T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(T_{\Pi_G}(\emptyset))$.

$N_\alpha = T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(T_{([\Pi_P]_{V_O^{\#n}})^N}(N_{\alpha-1}))$, where $1 \leq \alpha$.

Since $EHB(\Pi_O^{\#n})$ is a finite set and $[\Pi_O^{\#n}]^N = \Pi_G \cup ([\Pi_P]_{V_O^{\#n}})^N \cup [\Pi_O^{H_{\#n}}]^N$, it follows that there is $k \in \{0, 1, ...\}$ such that $N_k = N_{k+1} = N$.

Let $H(p, type, TotalProperty) \in N_0$, for $p \in V_O^{\#n}$. As $N_0 \subseteq N$, it follows that $H(p, type, TotalProperty) \in N$. Therefore, for all $x, y \in V_O^{\#n}$, $[\neg]H(x, p, y) \in N_0$ iff $[\neg]H(x, p, y) \in N$. Similarly, let $H(c, type, TotalClass) \in N_0$, for $c \in V_O^{\#n}$. Then, $H(c, type, TotalClass) \in N$. Therefore, for all $x \in V_O^{\#n}$, $[\neg]H(x, type, c) \in N_0$ iff $[\neg]H(x, type, c) \in N$. Now, from the above and as N_0 is the smallest subset of $EHB(\Pi_O^{\#n})$ that satisfies all rules in $\Pi_G \cup [\Pi_O^{H_{\#n}}]^N$, it follows that N_0 satisfies all rules in $[\Pi_O^{H_{\#n}}]$. Therefore, N_0 is a minimal subset of $EHB(\Pi_O^{\#n})$ that satisfies all rules $\Pi_G \cup [\Pi_O^{H_{\#n}}]$. Thus, $ELP^{-1}(N_0) \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \models sk(G)\})$.

Let α such that $1 \leq \alpha \leq k+1$. Note that N_α is the smallest subset of $EHB(\Pi_O^{\#n})$ such that (i) $N_\alpha \supseteq N_{\alpha-1}$, (ii) satisfies all rules in $[\Pi_O^{H_{\#}}]^N$, and (iii) $Head(r) \in N_\alpha$, for all $r \in ([\Pi_P]_{V_O^{\#n}})^N$ such that $Body(r) \subseteq N_{\alpha-1}$. Let $H(p, type, TotalProperty) \in N_\alpha$, for $p \in V_O^{\#n}$. As $N_\alpha \subseteq N$, it follows that $H(p, type, TotalProperty) \in N$. Therefore, for all $x, y \in V_O^{\#n}$, $[\neg]H(x, p, y) \in N_\alpha$ iff $[\neg]H(x, p, y) \in N$. Similarly, let $H(c, type, TotalClass) \in N_\alpha$, for $c \in V_O^{\#n}$. Then, $H(c, type, TotalClass) \in N$. Therefore, for all $x \in V_O^{\#n}$, $[\neg]H(x, type, c) \in N_\alpha$ iff $[\neg]H(x, type, c) \in N$. Now, from the above, it follows that N_α is a minimal subset of $EHB(\Pi_O^{\#n})$ such that (i) $N_\alpha \supseteq N_{\alpha-1}$, (ii) satisfies all rules in $[\Pi_O^{H_{\#n}}]$, and (iii) $Head(r) \in N_\alpha$, for all $r \in ([\Pi_P]_{V_O^{\#n}})^N$ such that $Body(r) \subseteq N_{\alpha-1}$. Therefore, as $N_{\alpha-1} \subseteq N$, it follows

that $ELP^{-1}(N_\alpha) \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \geq ELP^{-1}(N_{\alpha-1})$ and $I \models Concl(r)$, for all $r \in [P]_{V_O^{\#n}}$ s.t. $J \models Cond(r)$, $\forall J \in [ELP^{-1}(N_{\alpha-1}), ELP^{-1}(N)]\})$.

Now as $ELP^{-1}(N_0) \leq ... \leq ELP^{-1}(N_{k+1})$ and $ELP^{-1}(N_k) = ELP^{-1}(N_{k+1}) = ELP^{-1}(N) = M$, it follows that $M \in \mathcal{M}^{st_{\#n}}(O)$. □

Proposition 5 Let O be a simple ERDF ontology and let $n \geq n_O$. Let F be a simple ERDF formula over $V_O^{\#n}$.

1. If $\Pi_O^{\#n}$ is a non-contradictory ELP then $Ans_O^{st_{\#n}}(F) = Ans^{AS}_{\Pi_O^{\#n}}(L_F)$.
2. Otherwise, $\mathcal{M}^{st_{\#n}}(O) = \emptyset$.

Proof:
1.
$\Rightarrow$) Let $v \in Ans_O^{st_{\#n}}(F)$ and let $v(F) = t_1 \wedge ... \wedge t_k \wedge \sim t_{k+1} \wedge ... \wedge \sim t_n$, where $t_i = [\neg]p_i(s_i, o_i)$. We will show that $v \in Ans^{AS}_{\Pi_O^{\#n}}(L_F)$. Let N be a consistent answer set of $\Pi_O^{\#n}$. Then, based on Theorem 2, $ELP^{-1}(N) \in \mathcal{M}_O^{st_{\#n}}$. Thus, $ELP^{-1}(N) \models v(F)$. Then, $N \models L_{t_i}$, for $i = 1, ..., k$. Additionally, $N \models \sim L_{t_i}$, for $i = k+1, ..., n$. Thus, $N \models v(L_F)$. Therefore, $v \in Ans^{AS}_{\Pi_O^{\#n}}(L_F)$.

$\Leftarrow$) Let $v \in Ans^{AS}_{\Pi_O^{\#n}}(L_F)$ and let $v(F) = t_1 \wedge ... \wedge t_k \wedge \sim t_{k+1} \wedge ... \wedge \sim t_n$, where $t_i = [\neg]p_i(s_i, o_i)$. We will show that $v \in Ans_O^{st_{\#n}}(F)$. Let $M \in \mathcal{M}_O^{st_{\#n}}$. Then, based on Theorem 2, $ELP(M)$ is a consistent answer set of $\Pi_O^{\#n}$. Then, $ELP(M) \models v(L_F)$. Thus, $ELP(M) \models L_{t_i}$, for $i = 1, ..., k$. Additionally, $ELP(M) \models \sim L_{t_i}$, for $i = k+1, ..., n$. Therefore, $M \models t_i$, for $i = 1, ..., k$. Additionally, $M \models \sim t_i$, for $i = k+1, ..., n$. Thus, $M \models v(F)$. Thus, $v \in Ans_O^{st_{\#n}}(F)$.

2. If $\Pi_O^{\#n}$ is a contradictory ELP then there is no consistent answer set of $\Pi_O^{\#n}$. Assume now that there is an $M \in \mathcal{M}^{st_{\#n}}(O)$. Then, based on Theorem 2, $ELP(M)$ is a consistent answer set of $\Pi_O^{\#n}$, which is impossible. Therefore, $\mathcal{M}^{st_{\#n}}(O) = \emptyset$. □

Proposition 6 Let $D = (V, E)$ be a graph. D is *3-colorable* iff $O_D = \langle G_D, P_D \rangle$ has a $\#n$-stable model, for $n \in I\!N$.
Proof:
$\Rightarrow$) Assume that $D = (V, E)$ is *3-colorable* and let *color* be a mapping from V to {*Red, Green, Blue*}. Let I be the minimal $\#n$-Herbrand interpretation of O_D such that: $I \models G_D$ and

1. $PT_I(edge) = \{\langle v, u\rangle \mid (v, u) \in E\}$ and $PF_I(edge) = \emptyset$.
2. $PT_I(edgeR) = \{\langle v, u\rangle \mid (v, u) \in E$ and $color(v) = Red\}$ and
   $PF_I(edgeR) = Res_{O_D}^{H_{\#n}} \times Res_{O_D}^{H_{\#n}} - PT_I(edgeR)$.
3. $PT_I(edgeG) = \{\langle v, u\rangle \mid (v, u) \in E$ and $color(v) = Green\}$ and
   $PF_I(edgeG) = Res_{O_D}^{H_{\#n}} \times Res_{O_D}^{H_{\#n}} - PT_I(edgeG)$.
4. $PT_I(edgeB) = \{\langle v, u\rangle \mid (v, u) \in E$ and $color(v) = Blue\}$ and
   $PF_I(edgeB) = Res_{O_D}^{H_{\#n}} \times Res_{O_D}^{H_{\#n}} - PT_I(edgeB)$.

5. $CT_I(Red) = \{v \mid \exists (v, u) \in E$ and $color(v) = Red\}$.
6. $CT_I(Green) = \{v \mid \exists (v, u) \in E$ and $color(v) = Green\}$.
7. $CT_I(Blue) = \{v \mid \exists (v, u) \in E$ and $color(v) = Blue\}$.

8. $CT_I(NotRed) = CT_I(Green) \cup CT_I(Blue) \cup \{u \mid \exists (v, u) \in E$ and $color(v) = Red\}$.
9. $CT_I(NotGreen) = CT_I(Red) \cup CT_I(Blue) \cup \{u \mid \exists (v, u) \in E$ and $color(v) = Green\}$.

10. $CT_I(NotBlue) = CT_I(Red) \cup CT_I(Green) \cup \{u \mid \exists (v,u) \in E \text{ and } color(v) = Blue\}$.

Note that I does not satisfy the body of any constraint in P_D. It is easy the see that I is a $\#n$-stable model of O_D.

$\Leftarrow$) Let I be a $\#n$-stable model of O_D. We define:

1. $color(v) = Red$, for all $v \in V$ such that it does not exist $u \in V$ s.t. $\langle v,u\rangle \in PT_I(edge)$ or $\langle u,v\rangle \in PT_I(edge)$.

2. $color(v) = Red$, for all $v \in V$ s.t. it exists $u \in V$ s.t. $\langle v,u\rangle \in PT_I(edgeR)$.
3. $color(v) = Green$, for all $v \in V$ s.t. it exists $u \in V$, $\langle v,u\rangle \in PT_I(edgeG)$.
4. $color(v) = Blue$, for all $v \in V$ s.t. it exists $u \in V$, $\langle v,u\rangle \in PT_I(edgeB)$.

5. $color(u) = Blue$, for all $u \in V$ s.t. it exists $v \in V$, $\langle v,u\rangle \in PT_I(edgeR)$ and it does not exist $u' \in V$ s.t. $\langle u,u'\rangle \in PT_I(edgeG)$.
6. $color(u) = Green$, for all $u \in V$ s.t. it exists $v \in V$, $\langle v,u\rangle \in PT_I(edgeB)$ and it does not exist $u' \in V$ s.t. $\langle u,u'\rangle \in PT_I(edgeR)$.
7. $color(u) = Red$, for all $u \in V$ s.t. it exists $v \in V$, $\langle v,u\rangle \in PT_I(edgeG)$ and it does not exist $u' \in V$ s.t. $\langle u,u'\rangle \in PT_I(edgeB)$.

We will show that *color* is a mapping from the vertices V to colors $\{Red, Green, Blue\}$ such that if $(v,u) \in E$ then $color(v) \neq color(u)$.

If $v \in V$ s.t. there is no edge $\langle u,u'\rangle \in E$ with $v = u$ or $v = u'$ then v is assigned the color *Red* (due to rule 1). If $v \in V$ and there is an edge $\langle v,u\rangle \in E$ then v is assigned one of the colors *Red, Green, Blue* (due to rules 2-4). If (i) $v \in V$, (ii) v is not assigned a color due to rules 2-4, and (iii) there is an edge $\langle u,v\rangle \in E$ then v is assigned one of the colors *Red, Green, Blue* (due to rules 5-7). Note that, due to constraints in P_D, no $v \in V$ is assigned more than one color and that if $(v,u) \in E$ then $color(v) \neq color(u)$. Thus, graph $D = (V,E)$ is 3-colorable. □

Proposition 7 Let $O = \langle G,P\rangle$ be a bounded, simple ERDF ontology and let $n \geq n_O$. The problem of establishing whether O has a $\#n$-stable model is NP-complete w.r.t. the size of O.
Proof:
Hardness) It follows from Proposition 9.1 (*Hardness*), since a bounded objective ontology is a bounded simple ontology.

Membership) Guess now a $\#n$-semi-Herbrand interpretation M of O. Due to Theorem 2, it is the case that $M \in \mathcal{M}^{st_{\#n}}(O)$ iff $ELP(M)$ is a consistent answer set of $\Pi_O^{\#n}$. Thus, to verify that $M \in \mathcal{M}^{st_{\#n}}(O)$, it is enough to verify that $T^{\uparrow\omega}_{[\Pi_O^{\#n}]^{ELP(M)}}(\emptyset) = ELP(M)$. Note that the complexity of this step is polynomial w.r.t. the size of O, since O is a bounded, simple ERDF ontology. Thus, the complexity of deciding whether O has a $\#n$-stable model is in NP w.r.t. the size of O.

From the hardness and membership steps, above, it follows that the problem of establishing whether O has a $\#n$-stable model is NP-complete w.r.t. the size of O. □

Proposition 8 Let $O = \langle G,P\rangle$ be a bounded, simple ERDF ontology and let F be an ERDF formula. Additionally, let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \geq n_O$. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is co-NP-complete w.r.t. the size of O.
Proof:

Hardness) It follows from Proposition 9.3 (*Hardness*), since a bounded objective ontology is a bounded simple ontology.
Membership)
Case $max(\{i \in \mathbb{N} \mid rdf{:}\_i \in V_F\}) \leq n$: Assume that we want to verify if $O \not\models^{st_{\#n}} v(F)$. Guess now a $\#n$-semi-Herbrand interpretation I of O. Then, test if $I \in \mathcal{M}^{st_{\#n}}(O)$ (Step 1) as in the proof of the membership part of Proposition 7. If *Satisfies*(I, F) =FALSE (Step 2) then $O \not\models^{st_{\#n}} v(F)$. The complexity of Steps 1 and 2 are in P w.r.t. the size of O. Therefore, the complexity of checking whether $O \not\models^{st_{\#n}} v(F)$ is in NP w.r.t. the size of O. Thus, the complexity of deciding whether $O \models^{st_{\#n}} v(F)$ is in co-NP w.r.t. the size of O.

Case $max(\{i \in \mathbb{N} \mid rdf{:}\_i \in V_F\}) > n$: We want to verify if $O \models^{st_{\#n}} v(F)$. This is true only if O has no $\#n$-stable model. From Proposition 7, it follows that the complexity of this problem is in co-NP w.r.t. the size of O.

From the hardness and membership steps, above, it follows that the problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is co-NP-complete w.r.t. the size of O. □

Proposition 9 Let $O = \langle G, P \rangle$ be a bounded, objective ERDF ontology. Let G' be an ERDF graph, let F^d be an ERDF d-formula, and let F be an ERDF formula. Let $n \geq n_O$.

1. The problem of establishing whether O has a $\#n$-stable model is NP-complete w.r.t. the size of O.
2. The problems of establishing whether:
   (i) $O \models^{st_{\#n}} G'$, (ii) $O \models^{st_{\#n}} F^d$
   are co-NP-complete w.r.t. the size of O,
3. Let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) let a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \geq n_O$. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is co-NP-complete w.r.t. the size of O.

Proof:
1)
Hardness) Let $D = (V, E)$ be a graph. Note that $O_D = \langle G_D, P_D \rangle$ is a bounded, objective ERDF ontology and to generate O_D from D, it takes polynomial time w.r.t. the size of D. Now, from Proposition 6, and since *3-colorability* is an NP-complete problem, it follows that the problem of establishing whether O has a $\#n$-stable model is NP-hard w.r.t. the size of O.

Membership) Note that a bounded, objective ERDF ontology is also a bounded, simple ERDF ontology. Therefore, it follows directly from Proposition 7 that the problem of establishing whether O has a $\#n$-stable model is in NP w.r.t. the size of O.

2.i)
Hardness) Let $G' = \{p(s, o), \neg p(s, o)\}$, for $p, s, o \in \mathcal{URI}$. Then, $O \models^{st_{\#n}} G'$ iff O has no $\#n$-stable model. From Proposition 9.1, it follows that the complexity of deciding whether O has a $\#n$-stable model is NP-hard w.r.t. the size of O. Therefore, the complexity of deciding whether $O \models^{st_{\#n}} G'$ is co-NP-hard w.r.t. the size of O.

Membership) It is the case that $O \models^{st_{\#n}} G'$ iff "yes" $\in Ans_O^{st_{\#n}}(formula(G'))$. It follows from Proposition 8, that the complexity of answering if "yes" $\in Ans_O^{st_{\#n}}(formula(G'))$ is in co-NP w.r.t. the size of O. Therefore, the complexity of deciding if $O \models^{st_{\#n}} G'$ is in co-NP w.r.t. the size of O.

2.ii)
Hardness) Let $F^d = p(s,o) \wedge \neg p(s,o)$, for $p, s, o \in \mathcal{URI}$. Then, $O \models^{st_{\#n}} F^d$ iff O has no $\#n$-stable model. From Proposition 9.1, it follows that the complexity of deciding whether O has a $\#n$-stable model is NP-hard w.r.t. the size of O. Therefore, the complexity of deciding whether $O \models^{st_{\#n}} F^d$ is co-NP-hard w.r.t. the size of O.
Membership) It is the case that $O \models^{st_{\#n}} F^d$ iff "yes" $\in Ans_O^{st_{\#n}}(F^d)$. It follows from Proposition 8, that the complexity of answering if "yes" $\in Ans_O^{st_{\#n}}(F^d)$ is in co-NP w.r.t. the size of O. Therefore, the complexity of deciding if $O \models^{st_{\#n}} F^d$ is in co-NP w.r.t. the size of O.

3)
Hardness) It follows directly from Proposition 9.2.
Membership) It follows directly from Proposition 8. □

Proposition 10 Let $\mathcal{F} = \forall ?x_1 ... \forall ?x_k \exists ?y_1 ... \exists ?y_m\ c_1 \wedge ..... \wedge\ c_l$ be a 2-universal quantified boolean formula. $\mathcal{F}$ is invalid iff $O_{\mathcal{F}} = \langle G_{\mathcal{F}}, P_{\mathcal{F}} \rangle$ has a $\#n$-stable model, for $n \in I\!N$.
Proof:
$\Rightarrow$) Assume that $\mathcal{F}$ is invalid. Then, there exist truth values X_i, for $?x_i$, $i = 1, ..., k$, such that for all truth values for $?y_i$, for $i = 1, ..., m$, it is the case that $\sim(c_1 \wedge ..... \wedge c_l)$ is valid.

Let I_0 be the minimal $\#n$-Herbrand interpretation of $O_{\mathcal{F}}$ s.t. $I_0 \models G_{\mathcal{F}}$ and $x_i \in CT_{I_0}(IsTrue)$ if and only if $X_i = true$. If $X_i = true$, we say that $rdf{:}type(x_i, IsTrue)$ is evaluated to $true$ and If $X_i = false$, we say that $\neg rdf{:}type(x_i, IsTrue)$ is evaluated to $true$.

Let I_1 be the minimal $\#n$-Herbrand interpretation of $O_{\mathcal{F}}$ that extends I_0 s.t.:

1. If c_i has variables $?x_j$, $?x_{j'}$, and $?y_{j''}$ then
   $PT_{I_1}(c_i) = \{\langle x, x' \rangle \mid$ if $c_i(x, x')$ is the head of a rule in $P_{\mathcal{F}}$ whose body is evaluated to true $\}$.
2. If c_i has variables $?x_j$, $?y_{j'}$, and $?y_{j''}$ then
   $PT_{I_1}(c_{i,1}) = \{\langle x, x' \rangle \mid$ if $c_{i,1}(x, x')$ is the head of a rule in $P_{\mathcal{F}}$ whose body is evaluated to true $\}$, and
   $PT_{I_1}(c_{i,2}) = \{\langle x, x' \rangle \mid$ if $c_{i,2}(x, x')$ is the head of a rule in $P_{\mathcal{F}}$ whose body is evaluated to true $\}$.
3. If c_i has variables $?y_j$, $?y_{j'}$, and $?y_{j''}$ then
   $PT_{I_1}(c_{i,1}) = \{\langle x, x' \rangle \mid$ if $c_{i,1}(x, x')$ is the head of a rule in $P_{\mathcal{F}}\}$,
   $PT_{I_1}(c_{i,2}) = \{\langle x, x' \rangle \mid$ if $c_{i,2}(x, x')$ is the head of a rule in $P_{\mathcal{F}}\}$, and
   $PT_{I_1}(c_{i,3}) = \{\langle x, x' \rangle \mid$ if $c_{i,3}(x, x')$ is the head of a rule in $P_{\mathcal{F}}\}$.
4. If c_i has variables $?x_j$, $?x_{j'}$, and $?x_{j''}$ then
   $PT_{I_1}(c_i) = \{\langle s, s \rangle \mid$ if $c_i(s, s)$ is the head of a rule in $P_{\mathcal{F}}$ whose body is evaluated to true $\}$.

Note that since $\mathcal{F}$ is invalid, I_1 does not satisfy the body of the rule $w(s,s) \leftarrow c'_1, ..., c'_l.$ in $P_{\mathcal{F}}$. Thus, I_1 does not satisfy the body of the single constraint in $P_{\mathcal{F}}$. It is easy to see that I_1 is a $\#n$-stable model of $O_{\mathcal{F}}$.

$\Leftarrow$) Assume that I is a $\#n$-stable model of $O_{\mathcal{F}}$. Assume that we assign to each variable $?x_i$ the truth value $true$ if $x_i \in CT_I(IsTrue)$ and the truth value $false$ if $x_i \notin CT_I(IsTrue)$. Since I is a $\#n$-stable model of $O_{\mathcal{F}}$, $I \not\models w(s,s)$. This means that I does not satisfy the body of the rule $w(s,s) \leftarrow c'_1, ..., c'_l.$ in $P_{\mathcal{F}}$. From this it follows that for all truth values of the variables $?y_1, ..., ?y_m$, it holds $\sim(c_1 \wedge ... \wedge\ c_l)$, given the above thruth assignments of the variables $?x_1, ..., ?x_n$. Thus, $\mathcal{F}$ is invalid. □

Proposition 11 Let $O = \langle G, P\rangle$ be a simple ERDF ontology and let F be an ERDF formula. Additionally, let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \in I\!N$.

1. The problem of establishing whether O has a $\#n$-stable model is Σ_2^P =NP$^{\text{NP}}$-complete w.r.t. the size of O.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is Π_2^P =co-NP$^{\text{NP}}$-complete w.r.t. the size of O.

1.
Hardness) It follows from Proposition 12.1 (*Hardness*), since an objective ERDF ontology is a simple ERDF ontology.

Membership) Guess now a $\#n$-semi-Herbrand interpretation M of O. Due to Proposition 2, it is the case that $M \in \mathcal{M}^{st_{\#n}}(O)$ iff $ELP(M)$ is a consistent answer set of $\Pi_O^{\#n}$. Since the arities of the predicates appearing in $\Pi_O^{\#n}$ are bounded, it follows from Lemma 2 of [20], that deciding if $ELP(M)$ is a consistent answer set of $\Pi_O^{\#n}$ is in P$^{\text{NP}}$ w.r.t. the size of $\Pi_O^{\#n}$, and thus w.r.t. the size of O. Therefore, the complexity of deciding whether O has a $\#n$-stable model is in NP$^{\text{NP}}$ w.r.t. the size of O.

From the hardness and membership steps, above, it follows that the problem of establishing whether O has a $\#n$-stable model is NP$^{\text{NP}}$-complete w.r.t. the size of O.

2.
Hardness) It follows from Proposition 12.3 (*Hardness*), since an objective ERDF ontology is a simple ERDF ontology.

Membership)

Case $max(\{i \in I\!N \mid rdf{:}\_i \in V_F\}) \leq n$: Assume that we want to verify if $O \not\models^{st_{\#n}} v(F)$. Guess now a $\#n$-semi-Herbrand interpretation I of O. Then, test if $I \in \mathcal{M}^{st_{\#n}}(O)$ (Step 1) as in the proof of the membership part of Proposition 11.1. If *Satisfies*$(I, v(F))$ =FALSE (Step 2) then $O \not\models^{st_{\#n}} v(F)$. The complexity of Step 1 is in P$^{\text{NP}}$w.r.t. the size of O and the complexity of Step 2 is polynomial w.r.t. the size of O. Therefore, the complexity of checking whether $O \not\models^{st_{\#n}} v(F)$ is in NP$^{\text{NP}}$ w.r.t. the size of O. Thus, the complexity of deciding whether $O \models^{st_{\#n}} v(F)$ is in co-NP$^{\text{NP}}$ w.r.t. the size of O.

Case $max(\{i \in I\!N \mid rdf{:}\_i \in V_F\}) > n$: We want to verify if $O \models^{st_{\#n}} v(F)$. This is true only if O has no $\#n$-stable model. From Proposition 11.1, it follows that the complexity of this problem is in co-NP$^{\text{NP}}$ w.r.t. the size of O. □

Proposition 12 Let $O = \langle G, P\rangle$ be an objective ERDF ontology. Let G' be an ERDF graph, let F^d be an ERDF d-formula, and let F be an ERDF formula. Let $n \geq n_O$.

1. The problem of establishing whether O has a $\#n$-stable model is NP$^{\text{NP}}$-complete w.r.t. the size of O.
2. The problems of establishing whether:
   (i) $O \models^{st_{\#n}} G'$, (ii) $O \models^{st_{\#n}} F^d$
   are co-NP$^{\text{NP}}$-complete w.r.t. the size of O,

3. Let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) let a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \geq n_O$. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is co-NP$^{\text{NP}}$-complete w.r.t. the size of O. □

Proof:
1)
Hardness) Let $\mathcal{F}$ be a 2-universal quantified boolean formula. Note that $O_{\mathcal{F}} = \langle G_{\mathcal{F}}, P_{\mathcal{F}} \rangle$ is an ojective ERDF ontology and to generate $O_{\mathcal{F}}$ from $\mathcal{F}$, it takes polynomial time w.r.t. the size of $\mathcal{F}$. Now, from Proposition 10, and since the *2-QBF$_\forall$ problem* is a Π_2^P =co-NP$^{\text{NP}}$-complete problem, it follows that the problem of establishing whether O has a $\#n$-stable model is Σ_2^P =NP$^{\text{NP}}$-hard w.r.t. the size of O.

Membership) Note that an objective ERDF ontology is also a simple ERDF ontology. Therefore, it follows directly from Proposition 11 that the problem of establishing whether O has a $\#n$-stable model is in NP$^{\text{NP}}$ w.r.t. the size of O.

2.i)
Hardness) Let $G' = \{p(s,o), \neg p(s,o)\}$, for $p, s, o \in \mathcal{URI}$. Then, $O \models^{st_{\#n}} G'$ iff O has no $\#n$-stable model. From Proposition 12.1, it follows that the complexity of deciding whether O has a $\#n$-stable model is NP$^{\text{NP}}$-hard w.r.t. the size of O. Therefore, the complexity of deciding whether $O \models^{st_{\#n}} G'$ is co-NP$^{\text{NP}}$-hard w.r.t. the size of O.
Membership) It is the case that $O \models^{st_{\#n}} G'$ iff "yes" $\in Ans_O^{st_{\#n}}(formula(G'))$. It follows from Proposition 11, that the complexity of answering if "yes" $\in Ans_O^{st_{\#n}}(formula(G'))$ is in co-NP$^{\text{NP}}$ w.r.t. the size of O. Therefore, the complexity of deciding if $O \models^{st_{\#n}} G'$ is in co-NP$^{\text{NP}}$ w.r.t. the size of O.

2.ii)
Hardness) Let $F^d = p(s,o) \wedge \neg p(s,o)$, for $p, s, o \in \mathcal{URI}$. Then, $O \models^{st_{\#n}} F^d$ iff O has no $\#n$-stable model. From Proposition 12.1, it follows that the complexity of deciding whether O has a $\#n$-stable model is NP$^{\text{NP}}$-hard w.r.t. the size of O. Therefore, the complexity of deciding whether $O \models^{st_{\#n}} F^d$ is co-NP$^{\text{NP}}$-hard w.r.t. the size of O.
Membership) It is the case that $O \models^{st_{\#n}} F^d$ iff "yes" $\in Ans_O^{st_{\#n}}(F^d)$. It follows from Proposition 11, that the complexity of answering if "yes" $\in Ans_O^{st_{\#n}}(F^d)$ is in co-NP$^{\text{NP}}$ w.r.t. the size of O. Therefore, the complexity of deciding if $O \models^{st_{\#n}} F^d$ is in co-NP$^{\text{NP}}$ w.r.t. the size of O. □

3)
Hardness) It follows directly from Proposition 12.2.
Membership) It follows directly from Proposition 11.2. □

Theorem 3 Let O be a general ERDF ontology and let $n \geq n_O$. Let M be a $\#n$-semi-Herbrand-interpretation of O. It is the case that: $M \in \mathcal{M}^{st_{\#n}}(O)$ iff *Is-$\#n$-StableModelGeneral*(O, n, M)=TRUE.
Proof:
$\Rightarrow$) Let $M \in \mathcal{M}^{st_{\#n}}(O)$. We will show that *Is-$\#n$-StableModelGeneral*$(O, n, M)$=TRUE. Since $V_O^{\#n}$ is finite and $M \in \mathcal{M}^{st_{\#n}}(O)$, it follows that there is a sequence of $\#n$-Herbrand interpretations $I_0 \leq ... \leq I_{k+1}$ such that $I_k = I_{k+1} = M$ and:

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

2. For successor ordinals α with $0 < \alpha \leq k+1$:
$I_\alpha \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \geq I_{\alpha-1}$ and it is the case that:
$\forall\, r \in [P]_{V_O^{\#n}}$, if $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}$, then $I \models Concl(r)\}$.

Now, we define a sequence $N_0 \subseteq .... \subseteq N_{k+1} \subseteq EHB(\Pi_O^{\#n})$, as follows:

$N_0 = T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(T_{\Pi_G}(\emptyset))$.

$N_\alpha = T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(N_{\alpha-1} \cup \{L_{Concl(r)} \mid r \in [P]_{V_O^{\#n}}$ and $J \models Cond(r)$,
$\forall J \in [ELP^{-1}(N_{\alpha-1}), M]_O^{\#n}\})$, where $1 \leq \alpha \leq k+1$.

Lemma: It holds $N_\alpha = ELP(I_\alpha)$, for $\alpha = 0, ..., k+1$.
Proof: We will prove the Lemma, by induction.

First, we will show that $N_0 \subseteq ELP(I_0)$. Since $I_0 \models sk(G)$, it follows that $T_{\Pi_G}(\emptyset) \subseteq ELP(I_0)$. As $I_0 \in \mathcal{I}^{H_{\#n}}(O)$, it follows that $ELP(I_0)$ satisfies all rules in $[\Pi_O^{H_{\#n}}]$. Now, as $ELP(I_0) \subseteq N$, it follows that $ELP(I_0)$ satisfies all rules in $[\Pi_O^{H_{\#n}}]^N$. Moreover, as $T_{\Pi_G}(\emptyset) \subseteq ELP(I_0)$, it follows that $T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(T_{\Pi_G}(\emptyset)) \subseteq ELP(I_0)$. Therefore, $N_0 \subseteq ELP(I_0)$.

Let $H(p, type, TotalProperty) \in N_0$, for $p \in V_O^{\#n}$. As $N_0 \subseteq ELP(I_0) \subseteq ELP(M) = N$, it follows that $H(p, type, TotalProperty) \in N$. Therefore, for all $x, y \in V_O^{\#n}$, $[\neg]H(x, p, y) \in N_0$ iff $[\neg]H(x, p, y) \in N$. Similarly, let $H(c, type, TotalClass) \in N_0$, for $c \in V_O^{\#n}$. Then, $H(c, type, TotalClass) \in N$. Therefore, for all $x \in V_O^{\#n}$, $[\neg]H(x, type, c) \in N_0$ iff $[\neg]H(x, type, c) \in N$. Now, from the above and as N_0 satisfies all rules in $[\Pi_O^{H_{\#n}}]^N$, it follows that N_0 satisfies all rules in $[\Pi_O^{H_{\#n}}]$. Therefore, $ELP^{-1}(N_0)$ satisfies all semantic conditions of a (coherent) $\#n$-Herbrand interpretation of O. Thus, $ELP^{-1}(N_0) \in \mathcal{I}^{H_{\#n}}(O)$. Moreover, $ELP^{-1}(N_0) \models sk(G)$. Therefore, $ELP^{-1}(N_0) \in \{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \models sk(G)\}$. Now as $I_0 \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \models sk(G)\})$ and $N_0 \subseteq ELP(I_0)$, it follows that $N_0 = ELP(I_0)$.

Assumption: We assume that $N_{\alpha-1} = ELP(I_{\alpha-1})$, for an $\alpha \leq k$.

We will show that $N_\alpha = ELP(I_\alpha)$. First, we will show that $N_\alpha \subseteq ELP(I_\alpha)$. Due to assumption $N_{\alpha-1} = ELP(I_{\alpha-1})$ and the fact $I_{\alpha-1} \leq I_\alpha$, it follows that $N_{\alpha-1} \subseteq ELP(I_\alpha)$. Based on this and the fact $I_\alpha \models Concl(r)$, for all $r \in [P]_{V_O^{\#n}}$ s.t. $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}$, it follows that $N_{\alpha-1} \cup \{L_{Concl(r)} \mid r \in [P]_{V_O^{\#n}}$ and $J \models Cond(r), \forall J \in [ELP^{-1}(N_{\alpha-1}), M]_O^{\#n}\} \subseteq ELP(I_\alpha)$.

As $I_\alpha \in \mathcal{I}^{H_{\#n}}(O)$, it follows that $ELP(I_\alpha)$ satisfies all rules in $[\Pi_O^{H_{\#n}}]$. As $ELP(I_\alpha) \subseteq N$, it follows that $ELP(I_\alpha)$ satisfies all rules in $[\Pi_O^{H_{\#n}}]^N$. Now as $N_{\alpha-1} \cup \{L_{Concl(r)} \mid r \in [P]_{V_O^{\#n}}$ and $J \models Cond(r), \forall J \in [ELP^{-1}(N_{\alpha-1}), N]_O^{\#n}\} \subseteq ELP(I_\alpha)$, it follows that $T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(N_{\alpha-1} \cup \{L_{Concl(r)} \mid r \in [P]_{V_O^{\#n}}$ and $J \models Cond(r), \forall J \in [ELP^{-1}(N_{\alpha-1}), M]_O^{\#n}\}) \subseteq ELP(I_\alpha)$. Therefore, $N_\alpha \subseteq ELP(I_\alpha)$.

Let $H(p, type, TotalProperty) \in N_\alpha$, for $p \in V_O^{\#n}$. As $N_\alpha \subseteq ELP(I_\alpha) \subseteq ELP(M) = N$, it follows that $H(p, type, TotalProperty) \in N$. Therefore, for all $x, y \in V_O^{\#n}$, $[\neg]H(x, p, y) \in N_\alpha$ iff $[\neg]H(x, p, y) \in N$. Similarly, let $H(c, type, TotalClass) \in N_\alpha$, for $c \in V_O^{\#n}$. Then, $H(c, type, TotalClass) \in N$. Therefore, for all $x \in V_O^{\#n}$, $[\neg]H(x, type, c) \in N_\alpha$ iff $[\neg]H(x, type, c) \in N$. Now, from the above and as N_α satisfies

all rules in $[\Pi_O^{H_{\#n}}]^N$, it follows that N_α satisfies all rules in $[\Pi_O^{H_{\#n}}]$. Therefore, $ELP^{-1}(N_\alpha)$ satisfies all semantic conditions of a (coherent) $\#n$-Herbrand interpretation of O. Thus, $ELP^{-1}(N_\alpha) \in \mathcal{I}^{H_{\#n}}(O)$. Moreover, $ELP^{-1}(N_\alpha) \geq ELP^{-1}(N_{\alpha-1}) = I_{\alpha-1}$. Now, based on the assumption that $N_{\alpha-1} = ELP(I_{\alpha-1}) \subseteq N$ and the fact that $N_{\alpha-1} \cup \{L_{Concl(r)} \mid r \in [P]_{V_O^{\#n}}$ and $J \models Cond(r), \forall J \in [I_{\alpha-1}, M]_O^{\#n}\} \subseteq N_\alpha$, it follows that $ELP^{-1}(N_\alpha) \models Concl(r)$, for all $r \in [P]_{V_O^{\#n}}$ s.t. $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}$. Therefore, $ELP^{-1}(N_\alpha) \in \{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \geq I_{\alpha-1}$ and $I \models Concl(r)$, for all $r \in [P]_{V_O^{\#n}}$ s.t. $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}\}$. Now as $I_\alpha \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \geq I_{\alpha-1}$ and $I \models Concl(r)$, for all $r \in [P]_{V_O^{\#n}}$ s.t. $J \models Cond(r)$, $\forall J \in [I_{\alpha-1}, M]_O^{\#n}\})$ and $N_\alpha \subseteq ELP(I_\alpha)$, it follows that $N_\alpha = ELP(I_\alpha)$. *End of Lemma*

Therefore, $N_k = N_{k+1} = ELP(M)$. Thus, *Is-#n-StableModelGeneral*(O, n, M) =TRUE.

$\Leftarrow$) Let M be a $\#n$-semi-Herbrand-interpretation of O s.t. *Is-#n-StableModelGeneral*(O, n, M)=TRUE. We will show that $M \in \mathcal{M}^{st_{\#n}}(O)$.

We define a sequence $N_\alpha \subseteq EHB(\Pi_O^{\#n})$, $\alpha \in \{0, 1, ...\}$, as follows:

$$N_0 = T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(T_{\Pi_G}(\emptyset)).$$

$$N_\alpha = T^{\uparrow\omega}_{[\Pi_O^{H_{\#n}}]^N}(N_{\alpha-1} \cup \{L_{Concl(r)} \mid r \in [P]_{V_O^{\#n}} \text{ and } J \models Cond(r), \forall J \in [ELP^{-1}(N_{\alpha-1}), M]_O^{\#n}\}), \text{ where } 1 \leq \alpha.$$

Since *Is-#n-StableModelGeneral*(O, n, M)=TRUE, it follows that there is $k \in \{0, 1, ...\}$ such that $N_k = N_{k+1} = ELP(M)$.

Let $N = ELP(M)$. Let $H(p, type, TotalProperty) \in N_0$, for $p \in V_O^{\#n}$. As $N_0 \subseteq N$, it follows that $H(p, type, TotalProperty) \in N$. Therefore, for all $x, y \in V_O^{\#n}$, $[\neg]H(x, p, y) \in N_0$ iff $[\neg]H(x, p, y) \in N$. Similarly, let $H(c, type, TotalClass) \in N_0$, for $c \in V_O^{\#n}$. Then, $H(c, type, TotalClass) \in N$. Therefore, for all $x \in V_O^{\#n}$, $[\neg]H(x, type, c) \in N_0$ iff $[\neg]H(x, type, c) \in N$. Now, from the above and as N_0 is the smallest subset of $EHB(\Pi_O^{\#n})$ that satisfies all rules in $\Pi_G \cup [\Pi_O^{H_{\#n}}]^N$, it follows that N_0 satisfies all rules in $[\Pi_O^{H_{\#n}}]$. Therefore, N_0 is a minimal subset of $EHB(\Pi_O^{\#n})$ that satisfies all rules $\Pi_G \cup [\Pi_O^{H_{\#n}}]$. Thus, $ELP^{-1}(N_0) \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \models sk(G)\})$.

Let α such that $1 \leq \alpha \leq k+1$. Note that N_α is the smallest subset of $EHB(\Pi_O^{\#n})$ such that (i) $N_\alpha \supseteq N_{\alpha-1}$, (ii) satisfies all rules in $[\Pi_O^{H_{\#n}}]^N$, and (iii) $L_{Concl(r)} \in N_\alpha$, for all $r \in [P]_{V_O^{\#n}}$ and $J \models Cond(r)$, $\forall J \in [ELP^{-1}(N_{\alpha-1}), M]_O^{\#n}$. Let $H(p, type, TotalProperty) \in N_\alpha$, for $p \in V_O^{\#n}$. As $N_\alpha \subseteq N$, it follows that $H(p, type, TotalProperty) \in N$. Therefore, for all $x, y \in V_O^{\#n}$, $[\neg]H(x, p, y) \in N_\alpha$ iff $[\neg]H(x, p, y) \in N$. Similarly, let $H(c, type, TotalClass) \in N_\alpha$, for $c \in V_O^{\#n}$. Then, $H(c, type, TotalClass) \in N$. Therefore, for all $x \in V_O^{\#n}$, $[\neg]H(x, type, c) \in N_\alpha$ iff $[\neg]H(x, type, c) \in N$. Now, from the above, it follows that N_α is a minimal subset of $EHB(\Pi_O^{\#n})$ such that (i) $N_\alpha \supseteq N_{\alpha-1}$, (ii) satisfies all rules in $[\Pi_O^{H_{\#n}}]$, and (iii) $L_{Concl(r)} \in N_\alpha$, for all $r \in [P]_{V_O^{\#n}}$ and $J \models Cond(r)$, $\forall J \in [ELP^{-1}(N_{\alpha-1}), M]_O^{\#n}$. Therefore, it follows

that $ELP^{-1}(N_\alpha) \in minimal(\{I \in \mathcal{I}^{H_{\#n}}(O) \mid I \geq ELP^{-1}(N_{\alpha-1})$ and $I \models Concl(r)$, for all $r \in [P]_{V_O^{\#n}}$ s.t. $J \models Cond(r)$, $\forall J \in [ELP^{-1}(N_{\alpha-1}), M]\})$.

Now as $ELP^{-1}(N_0) \leq ... \leq ELP^{-1}(N_{k+1})$ and $ELP^{-1}(N_k) = ELP^{-1}(N_{k+1}) = M$, it follows that $M \in \mathcal{M}^{st_{\#n}}(O)$. □

Proposition 13 Let $F = \exists ?x_1 ... \exists ?x_k \forall ?y_1 ... \forall ?y_m R(?x_1, ..., ?x_k, ?y_1, ..., ?y_m)$ be a 2-quantified boolean formula. F is valid iff $O_F = \langle G_F, P_F \rangle$ has a $\#n$-stable model, for $n \in \mathbb{N}$.

Proof:

$\Rightarrow$)

Case $\forall ?y_1 ... \forall ?y_m R(false, ..., false, ?y_1, ..., ?y_m)$ is valid:

Let I_0 be the minimal $\#n$-Herbrand interpretation of O_F. Thus, $p_i, q_j, w \notin CT_{I_0}(rdf{:}Property)$, for $i = 1, ..., k$ and $j = 1, ..., m$. Thus, (i) $I_0 \models \sim p_i(s, s)$, for $i = 1, ..., k$, (ii) $I_0 \models \sim q_i(s, s)$, for $i = 1, ..., m$, and (iii) $I_0 \models \sim w(s, s)$.
Let I_1 be the minimal $\#n$-Herbrand interpretation of O_F s.t.:

1. $PF_{I_1}(p_i) = \{\langle s, s \rangle\}$ and $PT_{I_1}(p_i) = Res_{O_F}^{H_{\#n}} \times Res_{O_F}^{H_{\#n}} - PT_{I_1}(p_i)$, for $i = 1, ..., k$.
2. $PT_{I_1}(q_i) = \{\langle s, s \rangle\}$ and $PF_{I_1}(q_i) = \emptyset$, for $i = 1, ..., m$.
3. $PT_{I_1}(w) = \{\langle s, s \rangle\}$ and $PF_{I_1}(w) = \emptyset$.

It is easy to see that I_1 is a $\#n$-stable model of O_F generated by the sequence I_0, I_1.

Case $\forall ?y_1 ... \forall ?y_m R(false, ..., false, ?y_1, ..., ?y_m)$ is not valid:

Since $\exists ?x_1 ... \exists ?x_k \forall ?y_1 ... \forall ?y_m R(?x_1, ..., ?x_k, ?y_1, ..., ?y_m)$ is true, there exist truth values $X_1, ..., X_k$ such that $\forall ?y_1 ... \forall ?y_m R(X_1, ..., X_k, ?y_1, ..., ?y_m)$ is true,

As in the previous case, let I_0 be the minimal $\#n$-Herbrand interpretation of O_F. Let I_1 be the minimal $\#n$-Herbrand interpretation of O_F s.t.:

1. $PT_{I_1}(p_i) = \{\langle s, s \rangle\}$ and $PF_{I_1}(p_i) = Res_{O_F}^{H_{\#n}} \times Res_{O_F}^{H_{\#n}} - PT_{I_1}(p_i)$, if $X_i = true$, for $i = 1, ..., k$.
2. $PF_{I_1}(p_i) = \{\langle s, s \rangle\}$ and $PT_{I_1}(p_i) = Res_{O_F}^{H_{\#n}} \times Res_{O_F}^{H_{\#n}} - PF_{I_1}(p_i)$, if $X_i = false$, for $i = 1, ..., k$.

Let I_2 be the minimal $\#n$-Herbrand interpretation of O_F s.t.:

1. $PT_{I_2}(p_i) = \{\langle s, s \rangle\}$ and $PF_{I_2}(p_i) = Res_{O_F}^{H_{\#n}} \times Res_{O_F}^{H_{\#n}} - PT_{I_2}(p_i)$, if $X_i = true$, for $i = 1, ..., k$.
2. $PF_{I_2}(p_i) = \{\langle s, s \rangle\}$ and $PT_{I_2}(p_i) = Res_{O_F}^{H_{\#n}} \times Res_{O_F}^{H_{\#n}} - PF_{I_2}(p_i)$, if $X_i = false$, for $i = 1, ..., k$.
3. $PT_{I_2}(q_i) = \{\langle s, s \rangle\}$ and $PF_{I_2}(q_j) = \emptyset$, for $j = 1, ..., m$.
4. $PT_{I_2}(w) = \{\langle s, s \rangle\}$ and $PF_{I_2}(w) = \emptyset$.

It is easy to see that I_2 is a $\#n$-stable model of O_F generated by the sequence I_0, I_1, I_2.

$\Leftarrow$) Let $O_F = \langle G_F, P_F \rangle$ have a $\#n$-stable model M. Then, M is generated either by a sequence $I_0 < M$ or by a sequence $I_0 < I_1 < M$. Consider that M is generated by a sequence $I_0 < M$. Note that $I_0 \models \sim p_i(s, s)$, for $i = 1, ..., k$, (ii) $I_0 \models \sim q_j(s, s)$, for $j = 1, ..., m$, and (iii) $I_0 \models \sim w(s, s)$. Additionally note that due to the single constraint in P_F, it follows that $M \models w(s, s)$. Thus, $M \models q_j(s, s)$, for all $j = 1, ..., m$.

Further note that $M \models p_i(s,s) \vee \neg p_i(s,s)$, for $i = 1, ..., k$. Since M is a $\#n$-stable model of O_F, it follows that for all $J \in \mathcal{I}_{O_F}^{H_{\#n}}$ s.t. $I_0 \leq J \leq M$, it is the case that $J \models f_R$. It follows from this that F is valid.

Consider now that M is generated by a sequence $I_0 < I_1 < M$. Note that $I_0 \models \sim p_i(s,s)$, for $i = 1, ..., k$, (ii) $I_0 \models \sim q_j(s,s)$, for $j = 1, ..., m$, and (iii) $I_0 \models \sim w(s,s)$. Additionally, note that due to the single constraint in P_F, it follows that $M \models w(s,s)$. Thus, $M \models q_j(s,s)$, for all $j = 1, ..., m$. Further, note that (i) $I_1 \models p_i(s,s) \vee \neg p_i(s,s)$, for $i = 1, ..., k$, and (ii) $I_1 \models \sim q_j(s,s)$, for $j = 1, ..., m$. Since M is a $\#n$-stable model of O_F and $M \models w(s,s)$, it follows that for all $J \in \mathcal{I}_{O_F}^{H_{\#n}}$ s.t. $I_1 \leq J \leq M$, it is the case that $J \models f_R$. It follows from this that F is valid. □

Proposition 14 Let $O = \langle G, P \rangle$ be a bounded, ERDF ontology without quantifiers and let F be an ERDF formula. Additionally, let v be (i) "yes", 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 a $\#n$-stable model is Σ_2^P =NP$^{\text{NP}}$-complete w.r.t. the size of O.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is Π_2^P =co-NP$^{\text{NP}}$-complete w.r.t. the size of O.

Proof:
1.
Hardness) Let F be a 2-quantified boolean formula. Note that $O_F = \langle G_F, P_F \rangle$ is a bounded ERDF ontology without quantifiers and to generate O_F from F, it takes polynomial time w.r.t. the size of F. Now, from Proposition 13, and since the *2-QBF-problem* is a Σ_2^P =NP$^{\text{NP}}$-complete problem, it follows that the problem of establishing whether O has a $\#n$-stable model is Σ_2^P =NP$^{\text{NP}}$-hard w.r.t. the size of O.

Membership) It follows from Proposition 15.1 (*Membership*), since a bounded ERDF ontology without quantifiers is an ERDF ontology without quantifiers.

2.
Hardness) Let $F = p(s,o) \wedge \neg p(s,o)$, for $p, s, o \in \mathcal{URI}$. Then, $O \models^{st_{\#n}} F$ iff O has no $\#n$-stable model. From Proposition 14.1, it follows that the complexity of deciding whether O has a $\#n$-stable model is NP$^{\text{NP}}$-hard w.r.t. the size of O. Therefore, the complexity of deciding whether $O \models^{st_{\#n}} F$ is co-NP$^{\text{NP}}$-hard w.r.t. the size of O.
Membership) It follows from Proposition 15.2 (*Membership*), since a bounded ERDF ontology without quantifiers is an ERDF ontology without quantifiers.. □

Proposition 15 Let $O = \langle G, P \rangle$ be an ERDF ontology without quantifiers and let F be an ERDF formula. Additionally, let v be (i) "yes", 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 a $\#n$-stable model is Σ_2^P =NP$^{\text{NP}}$-complete w.r.t. the size of O.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is Π_2^P =co-NP$^{\text{NP}}$-complete w.r.t. the size of O.

Proof: First, we give a definition that will be used in the proof. Let Π be an ELP and let C be a set of constants. By $[\Pi]_C$, we denote the instantiation of Π w.r.t. the constants appearing in C.

1.
Hardness) This hardness part follows directly from the hardness part of Proposition 14.1.
Membership)
Guess now a $\#n$-semi-Herbrand interpretation M of O. It is the case that $M \in \mathcal{M}^{st_{\#n}}(O)$ iff the following Condition 1 and Condition 2 hold.
Condition 1) The following should hold:

1. For all $t \in sk(G)$, it is the case that $Satisfies(M, t)$=TRUE.
2. For all $r \in [P]_{V_O^{\#n}}$, if $Satisfies(M, Cond(r))$=TRUE then $Satisfies(M, Concl(r))$=TRUE.
3. For all $r \in [\Pi_O^{H_{\#n}}]_{V_O^{\#n}}$, it is the case that $ELP(M)$ satisfies r.

Condition 2) For each $L \in ELP(M)$, there is a founded proof $r_1\theta_1$, ...,$r_k\theta_k$ of L, where $r_i \in \Pi_G \cup P \cup \Pi_O^{H_{\#n}}$ and θ_i is a substitution of the free variables of r_i over $V_O^{\#n}$. We define:

$$L_i = \begin{cases} Head(r_i) & \text{if } r_i \in \Pi_G \\ L_{Concl(r_i\theta_i)} & \text{if } r_i \in P \\ Head(r_i\theta_i) & \text{if } r_i \in \Pi_O^{H_{\#n}} \end{cases}$$

Additionally, we define:

$$I_i = \begin{cases} ELP^{-1}(\{L_1, ..., L_{i-1}\}) & \text{if } r_i \in P \\ \{L_1, ..., L_{i-1}\} & \text{if } r_i \in \Pi_O^{H_{\#n}} \end{cases}$$

It should hold:

1. $L_k = L$.
2. If $r_i \in P$ then $MiddleNotSatisfies(O, n, I_i, M, Cond(r_i\theta_i))$=FALSE.
3. If $r_i \in \Pi_O^{H_{\#n}}$ then $Body^+(r_i\theta_i) \subseteq I_i$ and $Body^-(r_i\theta_i) \cap ELP(M) = \emptyset$.

First, we consider Condition 1. The complexity of Condition 1.1 is in P. To check the complement of Condition 1.2, guess an $r \in P$ and a substitution θ of the free variables of r over $V_O^{\#n}$. Then, check if $Satisfies(M, Cond(r\theta))$=TRUE and $Satisfies(M, Concl(r\theta))$=FALSE. The complexity of this ckeck is in P. Thus, the complexity of the complement of Condition 1.1. is in NP. Therefore, the complexity of Condition 1.2. is in co-NP. To check the complement of Condition 1.3, guess an $r \in \Pi_O^{H_{\#n}}$ and a substitution θ of the free variables of r over $V_O^{\#n}$. Then, check if $ELP(M)$ does not satisfy $r\theta$. The complexity of this check is in P. Thus, the complexity of the complement of Condition 1.3 is in NP. Therefore, the complexity of Condition 1.3 is in co-NP.

Now, we consider Condition 2. Note that the size of $ELP(M)$ is in P w.r.t. the size of O. Additionally, note that the size of each founded proof of an $L \in ELP(M)$ is in P w.r.t. the size of O. The complexity of Condition 2.2 is in NP. This is because the complexity of the steps (2-4) of Algorithm $MiddleNotSatisfies(O, n, I_i, M, Cond(r_i\theta_i))$ is in P. The complexity of Condition 2.3 is in P. Therefore, the complexity of Condition 2 is in P^{NP}.

Thus, Conditions 1 and 2 can be solved by P, NP, and P^{NP} oracle calls. Note that $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{P}^{\mathrm{NP}}$. Thus, the complexity of the problem of establishing whether O has a $\#n$-stable model is in $\mathrm{NP}^{\mathrm{P}^{\mathrm{NP}}}$=$\mathrm{NP}^{\mathrm{NP}}$ w.r.t. the size of O.

2.
Hardness) This hardness part follows directly from the hardness part of Proposition 14.2.
Membership)
Case $max(\{i \in I\!N \mid rdf{:}\_i \in V_F\}) \leq n$: Assume that we want to verify if $O \not\models^{st_{\#n}} v(F)$. Guess now a $\#n$-semi-Herbrand interpretation I of O. Then, test if $I \in \mathcal{M}^{st_{\#n}}(O)$ (Step 1) as in the proof of the membership part of Proposition 15.1. If *Satisfies*$(I, v(F))$ =FALSE (Step 2) then $O \not\models^{st_{\#n}} v(F)$. The complexity of Step 1 is in P^{NP}w.r.t. the size of O and the complexity of Step 2 is polynomial w.r.t. the size of O. Therefore, the complexity of checking whether $O \not\models^{st_{\#n}} v(F)$ is in $\mathrm{NP}^{\mathrm{NP}}$ w.r.t. the size of O. Thus, the complexity of deciding whether $O \models^{st_{\#n}} v(F)$ is in co-$\mathrm{NP}^{\mathrm{NP}}$ w.r.t. the size of O.

Case $max(\{i \in I\!N \mid rdf{:}\_i \in V_F\}) > n$: We want to verify if $O \models^{st_{\#n}} v(F)$. This is true only if O has no $\#n$-stable model. From Proposition 15.1, it follows that the complexity of this problem is in co-$\mathrm{NP}^{\mathrm{NP}}$ w.r.t. the size of O. □

Proposition 16 Let $\Phi = \exists ?x_1 \forall ?x_2 ... Q_k ?x_k R(?x_1, ..., ?x_k)$ be a fully quantified boolean formula. Φ is valid iff $O_\Phi = \langle G_\Phi, P_\Phi \rangle$ has a $\#n$-stable model, for $n \in I\!N$.
Proof:
$\Rightarrow$) Assume that Φ is valid. Let I_0 be the minimal $\#n$-Herbrand interpretation of O_Φ. Let I_1 be the minimal $\#n$-Herbrand interpretation of O_Φ s.t. $PT_{I_1}(p_i) = \{\langle s, s\rangle\}$ and $PF_{I_1}(p_i) = \emptyset$, for $i = 1, ..., k$. Let I_2 be the minimal $\#n$-Herbrand interpretation of O_Φ s.t.: $PT_{I_2}(p_i) = \{\langle s, s\rangle\}$ and $PF_{I_2}(p_i) = \emptyset$, for $i = 1, ..., k$, and $PT_{I_2}(w) = \{\langle s, s\rangle\}$ and $PF_{I_2}(w) = \emptyset$.

Note that since Φ is valid, for all $J \in \mathcal{I}_{O_\Phi}^{H_{\#n}}$ s.t. $I_1 \leq J \leq I_2$, it is the case that $J \models \phi_R$. Therefore, I_2 is a $\#n$-stable model of O_Φ generated by the sequence I_0, I_1, I_2.

$\Leftarrow$) Let $O_\Phi = \langle G_\Phi, P_\Phi \rangle$ have a $\#n$-stable model M. Due to the single constraint in P_Φ, it follows that $M \models w(s, s)$. Further, $M \models p_i(s, s)$, for all $i = 1, ..., k$ and $M \models {\sim}p_i(x, y)$, for all $i = 1, ..., k$ and $\langle x, y\rangle \in V_{O_\Phi}^{\#n} \times V_{O_\Phi}^{\#n} - \{\langle s, s\rangle\}$. Since M is a $\#n$-stable model of O_Φ, it follows that $M \models \phi_R$. Thus, Φ is valid. □

Proposition 17 Let $O = \langle G, P\rangle$ be a bounded ERDF ontology and let F be an ERDF formula. Additionally, let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \to V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \in I\!N$.

1. The problem of establishing whether O has a $\#n$-stable model is PSPACE-complete w.r.t. the size of O.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is PSPACE-complete w.r.t. the size of O.

Proof:
1.
Hardness) Let Φ be a fully quantified boolean formula. Note that $O_\Phi = \langle G_\Phi, P_\Phi\rangle$ is a bounded ERDF ontology and to generate O_Φ from Φ, it takes polynomial time w.r.t. Φ. Now, from Proposition 16, and since the *QBF-problem* is a PSPACE-complete problem, it follows that the problem of establishing whether O has a $\#n$-stable model is PSPACE-hard w.r.t. the size of O.

Membership) It follows from Proposition 18.1 (*Membership*), since a bounded ERDF ontology is a general ERDF ontology.

2.
Hardness) Let $F = p(s,o) \wedge \neg p(s,o)$, for $p, s, o \in \mathcal{URI}$. Then, $O \models^{st_{\#n}} F$ iff O has no $\#n$-stable model. From Proposition 17.1, it follows that the complexity of deciding whether O has a $\#n$-stable model is PSPACE-hard w.r.t. the size of O. Therefore, the complexity of deciding whether $O \models^{st_{\#n}} F$ is co-PSPACE-hard w.r.t. the size of O. But co-PSPACE=PSPACE.
Membership) It follows from Proposition 18.2 (*Membership*), since a bounded ERDF ontology is a general ERDF ontology.. □

Proposition 18 Let $O = \langle G, P \rangle$ be a general ERDF ontology and let F be an ERDF formula. Additionally, let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \in I\!N$.

1. The problem of establishing whether O has a $\#n$-stable model is PSPACE-complete w.r.t. the size of O.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is PSPACE-complete w.r.t. the size of O.

Proof:
1.
Hardness) This hardness part follows directly from the hardness part of Proposition 17.1.

Membership)
Guess now a $\#n$-semi-Herbrand interpretation M of O. It is the case that $M \in \mathcal{M}^{st_{\#n}}(O)$ iff Condition 1 and Condition 2 in the proof of the membership part of Proposition 15.1 hold.

First, we consider Condition 1. The complexity of Condition 1.1 is in P. To check the complement of Condition 1.2, guess an $r \in P$ and a substitution θ of the free variables of r over $V_O^{\#n}$. Then, check if $Satisfies(M, Cond(r\theta))$=TRUE and $Satisfies(M, Concl(r\theta))$=FALSE. The complexity of this check is in PSPACE (due to the possible quantifiers appearing in $Cond(r)$). Thus, the complexity of the complement of Condition 1.1. is in NPSPACE=PSPACE. Therefore, the complexity of Condition 1.2. is in co-PSPACE=PSPACE. To check the complement of Condition 1.3, guess an $r \in \Pi_O^{H_{\#n}}$ and a substitution θ of the free variables of r over $V_O^{\#n}$. Then, check if $ELP(M)$ does not satisfies $r\theta$. The complexity of this check is in P. Thus, the complexity of the complement of Condition 1.3 is in NP. Therefore, the complexity of Condition 1.3 is in co-NP $\subseteq$ PSPACE.

Now, we consider Condition 2. Note that the size of $ELP(M)$ is in P w.r.t. the size of O. Additionally, note that the size of each founded proof of an $L \in ELP(M)$ is in P w.r.t. the size of O. The complexity of Condition 2.2 is in NPSPACE. This is because the complexity of the steps (2-4) of Algorithm $MiddleNotSatisfies(O, n, I_i, M, Cond(r_i\theta_i))$ is in PSPACE (due to the possible quantifiers appearing in $Cond(r_i)$). The complexity of Condition 2.3 is in P. Therefore, the complexity of Condition 2 is in $\mathrm{P}^{\mathrm{NPSPACE}}$=PSPACE.

Thus, Conditions 1 and 2 can be solved by P and PSPACE oracle calls. Thus, the complexity of the problem of establishing whether O has a $\#n$-stable model is in $\mathrm{NP}^{\mathrm{PSPACE}}$ $\subseteq$PSPACE$^{\mathrm{PSPACE}}$=PSPACE w.r.t. the size of O.

2.
Hardness) This hardness part follows directly from the hardness part of Proposition 17.2.
Membership)
Case $max(\{i \in I\!N \mid rdf{:}\_i \in V_F\}) \leq n$: Assume that we want to verify if $O \not\models^{st_{\#n}} v(F)$. Guess now a $\#n$-semi-Herbrand interpretation I of O. Then, test if $I \in \mathcal{M}^{st_{\#n}}(O)$ (Step 1) as in the proof of the membership part of Proposition 18.1. If *Satisfies*$(I, v(F))$ =FALSE (Step 2) then $O \not\models^{st_{\#n}} v(F)$. The complexity of Step 1 is in $\mathrm{P}^{\mathrm{NPSPACE}} \subseteq$ PSPACE w.r.t. the size of O. The complexity of Step 2 is in P w.r.t. the size of O. Therefore, the complexity of checking whether $O \not\models^{st_{\#n}} v(F)$ is in $\mathrm{NP}^{\mathrm{PSPACE}} \subseteq$PSPACE w.r.t. the size of O. Thus, the complexity of deciding whether $O \models^{st_{\#n}} v(F)$ is in co-PSPACE=PSPACE w.r.t. the size of O.

Case $max(\{i \in I\!N \mid rdf{:}\_i \in V_F\}) > n$: We want to verify if $O \models^{st_{\#n}} v(F)$. This is true only if O has no $\#n$-stable model. From Proposition 18.1, it follows that the complexity of this problem is in co-PSPACE=PSPACE w.r.t. the size of O. □

Proposition 19 Let $O = \langle G, P\rangle$ be a general ERDF ontology, let F' be an ERDF formula without quantifiers, and let F an ERDF formula. Additionally, let v be (i) "yes", if $FVar(F) = \emptyset$, or (ii) a mapping $v : FVar(F) \rightarrow V_O^{\#n}$, if $FVar(F) \neq \emptyset$. Let $n \in I\!N$.

1. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F')$ has the same complexity w.r.t. the size of O and F as the complexity w.r.t. the size of O, for all kinds of ERDF ontologies considered in this work.
2. The problem of establishing whether $v \in Ans_O^{st_{\#n}}(F)$ is PSPACE-complete w.r.t. the size of O and F, for all kinds of ERDF ontologies considered in this work. □

Proof:
1. The proof is the same as the proof of the propositions 8, 9.3, 11.2, 12.3, 14.2, 15.2, 17.2, and 18.2, where we replace the phrase "w.r.t. the size of O" by "w.r.t. the size of O and F".
2.
Hardness) Let $\Phi = \exists ?x_1 \forall ?x_2 ... Q_k ?x_k R(?x_1, ..., ?x_k)$ be a fully quantified boolean formula, defined in Section 9. Let $s \in \mathcal{URI}$. We denote by $\mathcal{F}_R$ the ERDF formula $\mathcal{F}_R = \exists ?x_1 \forall ?x_2 ... Q_k ?x_k R(\ p_1(?x_1,\ s),\ ...,\ p_k(?x_k, s))$. Let $G_\Phi = \{p_i(s,s) \mid i \in \{1...k\}\}$, let $P_\Phi = \{\}$, and let $O_\Phi = \langle G_\Phi, P_\Phi\rangle$.
Lemma: Φ is valid iff $O_\Phi \models \mathcal{F}_R$.
Proof:
$\Rightarrow$) Let Φ be valid. Note that O_Φ has a single $\#n$-stable model s.t. $M \models p_i(s,s)$, for all $i = 1, ..., k$ and $M \models {\sim}p_i(x,y)$, for all $i = 1, ..., k$ and $\langle x, y\rangle \in V_{O_\Phi}^{\#n} \times V_{O_\Phi}^{\#n} - \{\langle s, s\rangle\}$. Note that $M \models \mathcal{F}_R$. Therefore, $O_\Phi \models \mathcal{F}_R$.
$\Leftarrow$) Assume that $O_\Phi \models \mathcal{F}_R$. Note that O_Φ has a single $\#n$-stable model s.t. $M \models p_i(s,s)$, for all $i = 1, ..., k$ and $M \models {\sim}p_i(x,y)$, for all $i = 1, ..., k$ and $\langle x, y\rangle \in V_{O_\Phi}^{\#n} \times V_{O_\Phi}^{\#n} - \{\langle s, s\rangle\}$. Since $O_\Phi \models \mathcal{F}_R$, it follows that $M \models \mathcal{F}_R$. Then, Φ is valid.
End of Lemma

Note that to generate O_Φ from Φ, it takes polynomial time w.r.t. Φ. Now, since the *QBF-problem* is a PSPACE-complete problem, it follows from Lemma that the problem of establishing whether "*yes*" $\in Ans_O^{st_{\#n}}(F')$ is PSPACE-hard w.r.t. the size of O and F'. Since O_Φ is a bounded, objective ERDF ontology, it follows that

problem of establishing whether $v \in Ans_O^{st_{\#n}}(F')$ is PSPACE-hard w.r.t. the size of O and F', for all kinds of ERDF ontologies considered in this work.
Membership)
Case $max(\{i \in I\!N \mid rdf{:}\_i \in V_F\}) \leq n$: Assume that we want to verify if $O \not\models^{st_{\#n}} v(F)$. Guess now a $\#n$-semi-Herbrand interpretation I of O. Then, test if $I \in \mathcal{M}^{st_{\#n}}(O)$ (Step 1) as in the proof of the membership part of Proposition 18.1. If *Satisfies*$(I, v(F))$ =FALSE (Step 2) then $O \not\models^{st_{\#n}} v(F)$. The complexity of Step 1 is in $\mathrm{P}^{\mathrm{NPSPACE}} \subseteq$ PSPACE w.r.t. the size of O and F. The complexity of Step 2 is in PSPACE w.r.t. the size of O and F, due to the possible quantifiers appearing in the formula F. Therefore, the complexity of checking whether $O \not\models^{st_{\#n}} v(F)$ is in $\mathrm{NP}^{\mathrm{PSPACE}} \subseteq$PSPACE w.r.t. the size of O and F. Thus, the complexity of deciding whether $O \models^{st_{\#n}} v(F)$ is in co-PSPACE=PSPACE w.r.t. the size of O and F. Since O is a general ERDF ontology, it follows that the problem of establishing whether $O \models^{st_{\#n}} v(F)$ is in PSPACE w.r.t. the size of O and F, for all kinds of ERDF ontologies considered in this work.

Case $max(\{i \in I\!N \mid rdf{:}\_i \in V_F\}) > n$: We want to verify if $O \models^{st_{\#n}} v(F)$. This is true only if O has no $\#n$-stable model. From Proposition 18.1, it follows that the complexity of this problem is in co-PSPACE=PSPACE w.r.t. the size of O and F. Since O is a general ERDF ontology, it follows that the problem of establishing whether $O \models^{st_{\#n}} v(F)$ is in PSPACE w.r.t. the size of O and F, for all kinds of ERDF ontologies considered in this work. □

Appendix B: Table of Symbols

List of Symbols	
Symbol	*Description*
V_G	the URI references and literals appearing in an ERDF graph G
$Cond(r)$	the condition of an ERDF rule r
$Concl(r)$	the conclusion of an ERDF rule r
$Var(r)$	the set of variables of an ERDF rule r
$FVar(r)$	the set of free variables of an ERDF rule r
V_P	the URI references and literals appearing in an ERDF program P
V_F	the URI references and literals appearing in an ERDF formula F
$sk(G)$	the skolemization of an ERDF graph G
$O = \langle G, P \rangle$	an ERDF ontology
V_O	$V_{sk(G)} \cup V_P \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$
Res_O^H	V_O with the well-typed literals substituted by their corresponding XML values
$\mathcal{I}^H(O)$	the set of Herbrand interpretations of O
$\mathcal{M}^{st}(O)$	the set of stable models of O
$\mathcal{V}_{RDF}^{\#n}$	$\mathcal{V}_{RDF} - \{rdf{:}\_i \mid i > n\}$
$V_O^{\#n}$	$V_O - \{rdf{:}\_i \mid i > n\}$
$Res_O^{H_{\#n}}$	$Res_O^H - \{rdf{:}\_i \mid i > n\}$
$\mathcal{I}^{H_{\#n}}(O)$	the set of $\#n$-Herbrand interpretations of O
$[I, J]_O^{\#n}$	$\{I' \in \mathcal{I}^{H_{\#n}}(O) \mid I \leq I' \leq J\}$
$\mathcal{M}^{st_{\#n}}(O)$	the set of $\#n$-stable models of O
$Ans_O^{st_{\#n}}(F)$	the set of $\#n$-stable answers of an ERDF formula F w.r.t. O
$\Pi_O^{\#n}$	$\Pi_G \cup \Pi_P \cup \Pi_O^{H_{\#n}}$
$ELP(I)$	$\{H(s,p,o) \mid s,p,o \in V_O^{\#n}$ and $\langle I(s), I(o) \rangle \in PT_I(p)\} \cup$ $\{\neg H(s,p,o) \mid s,p,o \in V_O^{\#n}$ and $\langle I(s), I(o) \rangle \in PF_I(p)\}$
$Head(r)$	L_0, where $r = L_0 \leftarrow L_1, ..., L_m, \sim L_{m+1}, ..., \sim L_n \in \Pi$
$Body(r)^+$	$\{L_1, ..., L_m\}$, where $r = L_0 \leftarrow L_1, ..., L_m, \sim L_{m+1}, ..., \sim L_n$
$Body(r)^-$	$\{L_{m+1}, ..., L_n\}$, where $r = L_0 \leftarrow L_1, ..., L_m, \sim L_{m+1}, ..., \sim L_n$
$Body(r)$	$Body(r)^+ \cup Body(r)^-$
Π^N	$\{Head(r) \leftarrow Body(r)^+ \mid r \in \Pi$ and $Body(r)^- \cap N = \emptyset\}$.
$T_\Pi(N)$	$N \cup \{Head(r) \mid r \in \Pi$ and $Body(r) \subseteq N\}$.

Table 2. Symbols and Description

Appendix C: 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 [32]. For simplicity, we have eliminated the namespace from the URIs in $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS}$.

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 $\mathcal{URI} \cup \mathcal{PL} \cup \mathcal{TL}$.

Definition 13 (Simple interpretation). A *simple interpretation* I of a vocabulary V consists of:

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

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

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

Definition 14 (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: $p \in V \cap \mathcal{URI},\ s, o \in V \cup Var,\ I(p) \in Prop_I$, and $\langle [I + v](s), [I + v](o) \rangle \in PT_I(I(p))$.
- I *satisfies* the RDF graph G, denoted by $I \models G$, iff there exists a mapping $v : Var(G) \rightarrow Res_I$ such that $I, v \models G$. □

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

Table 3. The RDF axiomatic triples

Definition 15 (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 $\langle x, I(Property)\rangle \in PT_I(I(type))$.
2. If $s \text{ ^^}rdf{:}XMLLiteral \in V$ and s is a well-typed XML literal string, then
   $IL_I(s \text{ ^^}rdf{:}XMLLiteral)$ is the XML value of s,
   $IL_I(s \text{ ^^}rdf{:}XMLLiteral) \in LV_I$, and
   $\langle IL_I(s \text{ ^^}rdf{:}XMLLiteral), I(XMLLiteral)\rangle \in PT_I(I(type))$.
3. If $s \text{ ^^}rdf{:}XMLLiteral \in V$ and s is an ill-typed XML literal string then
   $IL_I(s \text{ ^^}rdf{:}XMLLiteral) \in Res_I - LV_I$, and
   $\langle IL_I(s \text{ ^^}rdf{:}XMLLiteral), I(XMLLiteral)\rangle \notin PT_I(I(type))$.
4. I satisfies the RDF axiomatic triples, shown in Table 3. □

Definition 16 (RDF entailment). Let G, G' be RDF graphs. We say that G *RDF-entails* G' ($G \models^{RDF} G'$) iff for every RDF interpretation I, if $I \models G$ then $I \models G'$. □

Definition 17 (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 $CT_I : Cls_I \to \mathcal{P}(Res_I)$, such that:

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

Definition 18 (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 $I \models G'$. □

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

Table 4. The RDFS axiomatic triples

References

1. J. J. Alferes, C. V. Damásio, and L. M. Pereira. Semantic Web Logic Programming Tools. In *International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR-2003)*, pages 16–32, 2003.
2. A. Analyti, G. Antoniou, and C. V. Damásio. A Formal Theory for Modular ERDF Ontologies. In *3rd International Conference Web Reasoning and Rule Systems (RR-2009)*, pages 212–226, 2009.
3. A. Analyti, G. Antoniou, and C. V. Damásio. MWeb: A Principled Framework for Modular Web Rule Bases and its Semantics. *ACM Transactions on Computational Logic*, 12(2), Article 17, 2011.
4. A. Analyti, G. Antoniou, C. V. Damasio, and I. Pachoulakis. A Framework for Modular ERDF Ontologies. *Annals of Mathematics and Artificial Intelligence*, 67(3-4):189–249, 2013.
5. A. Analyti, G. Antoniou, C. V. Damásio, and G. Wagner. Negation and Negative Information in the W3C Resource Description Framework. *Annals of Mathematics, Computing & Teleinformatics (AMCT)*, 1(2):25–34, 2004.
6. 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.
7. A. Analyti, G. Antoniou, C. V. Damásio, and G. Wagner. On the Computability and Complexity Issues of Extended RDF. In *10th Pacific Rim International Conference on Artificial Intelligence (PRICAI-2008)*, pages 5–16, 2008.
8. G. Antoniou, A. Bikakis, and G. Wagner. A System for Nonmonotonic Rules on the Web. In *3rd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RULEML-2003)*, pages 23–36, 2004.
9. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.
10. N. Bassiliades, G. Antoniou, and I. P. Vlahavas. DR-DEVICE: A Defeasible Logic System for the Semantic Web. In *2nd International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR-2004)*, pages 134–148, 2004.
11. R. Berger. The Undecidability of the Dominoe Problem. *Memoirs of the American Mathematical Society*, 66:1–72, 1966.
12. T. Berners-Lee. Design Issues - Architectual and Philosophical Points. Personal notes, 1998. Available at `http://www.w3.org/DesignIssues`.
13. 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.
14. G. Brewka, T. Eiter, and M. Truszczynski. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.
15. F. Bry, C. Ley, B. Linse, and B. Marnette. RDFLog: It's like Datalog for RDF. In *22nd Workshop on (Constraint) Logic Programming (WLP-2008), co-located with JELIA-2008*, 2008.
16. W. Chen, M. Kifer, and D. S. Warren. HILOG: A Foundation for Higher-Order Logic Programming. *Journal of Logic Programming*, 15(3):187–230, 1993.
17. C. V. Damásio, A. Analyti, and G. Antoniou. Embeddings of Simple Modular Extended RDF. In P. Hitzler and T. Lukasiewicz, editors, *4th International Conference on Web Reasoning and Rule Systems (RR-2010)*, pages 204–212, 2010.
18. F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.
19. W. Drabent and J. Maluszynski. Well-Founded Semantics for Hybrid Rules. In *First International Conference on Web Reasoning and Rule Systems (RR-2007)*, pages 1–15, 2007.

20. T. Eiter, W. Faber, M. Fink, and S. Woltran. Complexity Results for Answer Set Programming with Bounded Predicate Arities and Implications. *Annal of Mathematics and Artificial Intelligence*, 51(2-4):123–165, 2007.
21. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. In *9th International Conference on Principles of Knowledge Representation and Reasoning (KR-2004)*, pages 141–151, 2004.
22. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-Founded Semantics for Description Logic Programs in the Semantic Web. In *3rd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML-2004)*, pages 81–97, 2004.
23. P. Ferraris, J. Lee, and V. Lifschitz. Stable models and Circumscription. *Artificial Intelligence*, 175(1):236–263, 2011.
24. T. Furche. Exploring and Taming Existence in Rule-based RDF Queries. REWERSE Deliverable I4-D14, 2007. Available at `http://rewerse.net/deliverables/m42/i4-d14.pdf`.
25. M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012.
26. M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. Conflict-Driven Answer Set Solving. In *20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 386–392, 2007.
27. A. V. Gelder, K. A. Ross, and J. S. Schlipf. Unfounded Sets and Well-Founded Semantics for General Logic Programs. In *17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS-1988)*, pages 221–230, 1988.
28. A. V. Gelder, K. A. Ross, and J. S. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM*, 38(3):620–650, 1991.
29. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *5th International Conference on Logic Programming (ICLP-1988)*, pages 1070–1080, 1988.
30. M. Gelfond and V. Lifschitz. Logic programs with Classical Negation. In *7th International Conference on Logic Programming*, pages 579–597, 1990.
31. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9(3/4):365–386, 1991.
32. P. Hayes. RDF Semantics. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`.
33. 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.
34. P. Hitzler, M. Krotzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. OWL 2 Web Ontology Language Primer (Second Edition), 2012. W3C Recommendation 11 December 2012. Available at `http://www.w3.org/TR/owl2-primer/`.
35. I. Horrocks, O. Kutz, and U. Sattler. The Even More Irresistible SROIQ. In *10th International Conference on Principles of Knowledge Representation and Reasoning (KR-2006)*, pages 57–67, 2006.
36. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission, 21 May 2004. Available at `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`.
37. G. Ianni, A. Martello, C. Panetta, and G. Terracina. Efficiently Querying RDF(S) Ontologies with Answer Set Programming. *Journal of Logic and Computation*, 19(4):671–695, 2009.
38. G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

39. M. Knorr, J. J. Alferes, and P. Hitzler. A Coherent Well-founded Model for Hybrid MKNF Knowledge Bases. In *18th European Conference on Artificial Intelligence (ECAI-2008)*, pages 99–103, 2008.
40. M. Krötzsch, S. Rudolph, and P. Hitzler. Description Logic Rules. In *18th European Conference on Artificial Intelligence (ECAI-2008)*, pages 80–84, 2008.
41. J. Lee and Y. Meng. First-Order Stable Model Semantics and First-Order Loop Formulas. *Journal Artificial Intelligence Research (JAIR)*, 42:125–180, 2011.
42. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.
43. A. Y. Levy and M. Rousset. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.
44. V. Lifschitz. Nonmonotonic databases and epistemic queries. In *12th International Joint Conference on Artificial Intelligence (IJCAI-1991)*, pages 381–386, 1991.
45. V. Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138(1-2):39–54, 2002.
46. V. Lifschitz. What Is Answer Set Programming? In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, (AAAI-2008)*, pages 1594–1597, 2008.
47. J. W. Lloyd. *Foundations of Logic Programming (2nd edition)*. Springer-Verlag, 1987.
48. T. Lukasiewicz. A Novel Combination of Answer Set Programming with Description Logics for the Semantic Web. In *4th European Semantic Web Conference (ESWC-2007)*, pages 384–398, 2007.
49. V. Marek and M. Truszczyski. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer-Verlag, 1999.
50. J. McCarthy. Applications of Circumscription to Formalizing Common-Sense Knowledge. *Artificial Intelligence*, 28(1):89–116, 1986.
51. J. Mei, Z. Lin, and H. Boley. $\mathcal{ALC}^{u}_{\mathbb{F}}$: An Integration of Description Logic and General Rules. In *First International Conference on Web Reasoning and Rule Systems (RR-2007)*, pages 163–177, 2007.
52. B. Motik and R. Rosati. A Faithful Integration of Description Logics with Logic Programming. In *20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 477–482, 2007.
53. B. Motik and R. Rosati. Reconciling description logics and rules. *Journal of the ACM*, 57(5), 2010.
54. B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. In *3rd International Semantic Web Conference (ISWC-2004)*, pages 549–563, 2004.
55. S. Muñoz, J. Pérez, and C. Gutiérrez. Minimal Deductive Systems for RDF. In *4th European Semantic Web Conference (ESWC-2007)*, pages 53–67, 2007.
56. I. Niemelä. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999.
57. I. Niemelä and P. Simons. Smodels - An Implementation of the Stable Model and Well-Founded Semantics for Normal LP. In *4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-1997)*, pages 421–430, 1997.
58. C. M. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
59. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Transanctions on Database Systems*, 34(3), Article 16 (45 pages), 2009.
60. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, 15 January 2008. Available at `http://www.w3.org/TR/rdf-sparql-query/`.
61. R. Rosati. Towards Expressive KR Systems Integrating Datalog and Description Logics: Preliminary Report. In *Proc. of the 1999 Description Logic Workshop (DL-1999)*, pages 160–164, 1999.
62. R. Rosati. On the Decidability and Complexity of Integrating Ontologies and Rules. *Journal of Web Semantics*, 3:61–73, 2005.

63. R. Rosati. DL+log: Tight Integration of Description Logics and Disjunctive Datalog. In *10th International Conference on Principles of Knowledge Representation and Reasoning (KR-2006)*, 2006.
64. S. Schenk and S. Staab. Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the web. In *17th International Conference on World Wide Web (WWW-2008)*, pages 585–594, 2008.
65. M. Sintek and S. Decker. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *1st International Semantic Web Conference (ISWC-2002)*, pages 364–378. Springer-Verlag, 2002.
66. L. J. Stockmeyer and A. R. Meyer. Word Problems Requiring Exponential Time: Preliminary Report. In *Fifth Annual ACM Symposium on Theory of Computing (STOC-1973)*, pages 1–9, 1973.
67. H. J. ter Horst. Extending the RDFS Entailment Lemma. In *3rd International Semantic Web Conference (ISWC-2004)*, pages 77–91, 2004.
68. H. J. ter Horst. Completeness, Decidability and Complexity of Entailment for RDF Schema and a Semantic Extension Involving the OWL Vocabulary. *Journal of Web Semantics*, 3(2-3):79–115, 2005.
69. F. Yang and X. Chen. $\mathcal{DL}clog$: A Hybrid System Integrating Rules and Description Logics with Circumscription. In *2007 International Workshop on Description Logics (DL-2007)*, 2007.
70. G. Yang, M. Kifer, and C. Zhao. Flora-2: A Rule-Based Knowledge Representation and Inference Infrastructure for the Semantic Web. In *2nd International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE'03)*, pages 671–688, 2003.