Computability and Complexity Issues of Extended RDF

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

Abstract. ERDF stable model semantics is a recently proposed semantics for ERDF ontologies and a faithful extension of RDFS semantics on RDF graphs. Unfortunately, ERDF stable model semantics is in general undecidable. In this paper, we elaborate on the computability and complexity issues of the ERDF stable model semantics.

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 [1], the Semantic Web language RDFS [4] is extended to accommodate the two negations of Partial Logic [5], namely *weak negation* $\sim$ (expressing negation-as-failure or non-truth) and *strong negation* $\neg$ (expressing explicit negative information or falsity), as well as derivation rules. The new language is called *Extended RDF* ($ERDF$).

In [1], the *stable model semantics* of ERDF ontologies is developed, based on Partial Logic, extending the model-theoretic semantics of RDFS. 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.

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

In this paper, we elaborate on the computability and complexity issues of the ERDF stable model semantics. Additionally, we propose a slightly modified semantics on ERDF ontologies, called *ERDF #n-stable model semantics* that is also a faithful extension of RDFS semantics on RDF graphs and achieves decidability.

2 Stable Model Semantics of ERDF Ontologies

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

A (Web) *vocabulary* V is a set of URI references and/or literals (plain or typed). We denote the set of all URI references by $\mathcal{URI}$. We consider a set of variable symbols Var such that $\mathcal{URI}$, Var, and the set of literals are pairwise disjoint. In our examples, variable symbols are prefixed by "?".

Let V be a vocabulary. An *ERDF triple* over V is an expression of the form $p(s,o)$ or $\neg p(s,o)$, where $s,o \in V \cup Var$ are called *subject* and *object*, respectively, and $p \in V \cap \mathcal{URI}$ is called *property*. An *ERDF graph* G is a set of ERDF triples over some vocabulary V. We denote the variables appearing in G by $Var(G)$, and the set of URI references and literals appearing in G by V_G.

Let V be a vocabulary. We denote by $L(V)$ the smallest set that contains the ERDF triples over V and is closed with respect to the following conditions: if $F, G \in L(V)$ then $\{\sim F,\ F \wedge G,\ F \vee G,\ F \supset G,\ \exists x F,\ \forall x F\} \subseteq L(V)$, where $x \in Var$. An *ERDF formula* over V is an element of $L(V)$. Intuitively, an ERDF graph G represents an existentially quantified conjunction of ERDF triples. Specifically, let $G = \{t_1, ..., t_m\}$ be an ERDF graph, and let $Var(G) = \{x_1, ..., x_k\}$. Then, G represents the ERDF formula $formula(G) = \exists ?x_1, ..., \exists ?x_k\ t_1 \wedge ... \wedge t_m$.

Existentially quantified variables in ERDF graphs are handled by $skolemization$. Let G be an ERDF graph. The *skolemization function* of G is an 1:1 mapping $sk_G : Var(G) \rightarrow \mathcal{URI}$, where for each $x \in Var(G)$, $sk_G(x)$ is an artificial URI, denoted by $G\!:\!x$. The *skolemization* of G, denoted by $sk(G)$, is the ground ERDF graph derived from G after replacing each $x \in Var(G)$ by $sk_G(x)$.

An *ERDF rule* r over a vocabulary V is an expression of the form: $Concl(r) \leftarrow Cond(r)$, where $Cond(r) \in L(V) \cup \{true\}$ and $Concl(r)$ is an ERDF triple or $false$. An *ERDF program* is a set of ERDF rules. We denote the set of URI references and literals appearing in P by V_P. An *ERDF ontology* is a pair $O = \langle G, P \rangle$, where G is an ERDF graph and P is an ERDF program.

The vocabulary of RDF, $\mathcal{V}_{RDF}$, is a set of $\mathcal{URI}$ references in the rdf: namespace [4]. The vocabulary of RDFS, $\mathcal{V}_{RDFS}$, is a set of $\mathcal{URI}$ references in the $rdfs$: namespace [4]. 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, at the interpretation level, each statement $rdf\!:\!type(x,c)$ is either true or explicitly false. Similarly, instances of the metaclass $erdf\!:\!TotalProperty$ are properties p that satisfy totalness, meaning that, at the interpretation level, each statement $p(x,y)$ is either true or explicitly false.

Let $O = \langle G, P \rangle$ be an ERDF ontology. The *vocabulary* of O is defined as $V_O = V_{sk(G)} \cup V_P \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. In [1], the set of *(ERDF) stable models* of O is defined, denoted by $\mathcal{M}^{st}(O)$. Each stable model M of O (i) interprets the terms in V_O and (ii) assigns intended truth and falsity extensions to the classes and properties in V_O (satisfying all semantic conditions of an RDFS interpretation [4] on V_O, as well as new semantic conditions, particular to ERDF). M is generated through a sequence of steps. Intuitively, starting of an intended interpretation for $sk(G)$, a stratified sequence of rule applications is produced, where all applied rules remain applicable throughout the generation of stable model M.

Let $M \in \mathcal{M}^{st}(O)$ and let F be an ERDF formula or ERDF graph. In [1], the *model relation* $M \models F$ is defined. We say that O *entails* F under the *(ERDF) stable model semantics*, denoted by $O \models^{st} F$, iff for all $M \in \mathcal{M}^{st}(O)$, $M \models F$.

As an example, consider a class $ex\!:\!Wine$ whose instances are wines and a property $ex\!:\!likes(X,Y)$ indicating that person X likes wine Y. Assume now that we want to select wines for a dinner such

[1] Institute of Computer Science, FORTH-ICS, Crete, Greece, e-mail: analyti@ics.forth.gr
[2] Department of Computer Science, University of Crete, Greece
[3] CENTRIA, Departamento de Informatica, Faculdade de Ciencias e Tecnologia, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
[4] Inst. of Informatics, Brandenburg Univ. of Technology at Cottbus, Germany

that for each guest, there is on the table exactly one wine that she/he likes. Let the class $ex{:}Guest$ indicate the persons that will be invited to the dinner and let the class ex:*SelectedWine* indicate the wines *chosen* to be served. An ERDF program P that describes this wine selection problem is the following[5,6]:

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

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

Then, the ERDF ontology $O = \langle G, P\rangle$ has *only* one stable model M, for which 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* $Retsina$. Obviously, $O \models^{st} rdf{:}type(Retsina,$ $SelectedWine) \wedge {\sim}rdf{:}type(Riesling, SelectedWine)$.

Proposition 2.1 Let G, G' be RDF graphs such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$, $V_{G'} \cap \mathcal{V}_{ERDF} = \emptyset$, and $V_{G'} \cap sk_G(Var(G)) = \emptyset$. It holds: $G \models^{RDFS} G'$ iff $\langle G, \emptyset\rangle \models^{st} G'$.

3 Computability and Complexity Issues

In [1], it is shown that satisfiability and entailment under the ERDF stable model semantics are in general undecidable. The proof of undecidability exploits a reduction from the *unbounded tiling problem*, whose existence of a solution is known to be undecidable [2]. Note that since each constraint $false \leftarrow F$ that appears in an ERDF ontology O can be replaced by the rule $\neg t \leftarrow F$, where t is an RDF, RDFS, or ERDF axiomatic triple, the presence of constraints in O does not affect decidability.

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

Reduction in [1] shows that ERDF stable model satisfiability and entailment remain undecidable, even if (i) $O = \langle G, P\rangle$ is a simple ERDF ontology, and (ii) the terms $erdf{:}TotalClass$ and $erdf{:}TotalProperty$ do not appear in O (i.e., $(V_G \cup V_P) \cap \mathcal{V}_{ERDF} = \emptyset$). However, we will show that satisfiability and entailment under the ERDF stable model semantics are decidable, if (i) O is an objective ERDF ontology, and (ii) the entailed formula is an 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 $F = (\exists ?x\ rdf{:}type(?x,\ Vertex) \wedge rdf{:}type(?x, Red))$ $\vee\ (\exists ?x\ rdf{:}type(?x,\ Vertex) \wedge \neg rdf{:}type(?x, Blue))$. Then, F is an ERDF *d*-formula. It is easy to see that if G is an ERDF graph then $formula(G)$ is an ERDF *d*-formula.

Let $O = \langle G, P\rangle$ be an ERDF ontology and let $n \in I\!N$, we define $V_O^{\#n} = V_O - \{rdf{:}\_i \mid i > n\}$. Additionally, we define: (i) $n_O = 0$, if $(V_G \cup V_P) \cap \{rdf{:}\_i \mid i \geq 1\} = \emptyset$, and (ii) $n_O = max(\{i \in I\!N \mid rdf{:}\_i \in V_G \cup V_P\})$, otherwise.

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

Proposition 3.1 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-complete w.r.t. size of $sk(G) \cup [P]_{V_O^{\#n_O}}$.
2. The problems of establishing whether:
(i) $O \models^{st} G'$ and (ii) $O \models^{st} F^d$ are co-NP-complete w.r.t. size of $sk(G) \cup [P]_{V_O^{\#n_O}}$.

The hardness part of the above complexity results can be proved by a reduction from the *Graph 3-Colorability* problem, which is a classical NP-complete problem. Moreover, participation of the above problems in NP or co-NP can be proved by showing that, from the infinite set of $rdf{:}\_i$ terms ($i \in I\!N$), only a finite subset needs to be considered for solving the corresponding problem.

The following proposition shows 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. This result can also be proved by a reduction from the unbounded tiling problem [2].

Proposition 3.2 Let G be an ERDF graph, let P be an objective program, and let F be an ERDF formula. The problem of establishing whether $\langle G, P\rangle \models^{st} F$ is in general undecidable.

Let O be a *general* ERDF ontology (with weak negation possibly appearing in the program rules). The source of undecidability of the ERDF stable model semantics of O is the fact that $\mathcal{V}_{RDF}$ is infinite. Thus, the vocabulary of O is also infinite (note that $\{rdf{:}\_i \mid i \geq 1\} \subseteq \mathcal{V}_{RDF} \subseteq V_O$). Therefore, we slightly modify the definition of the ERDF stable model semantics, based on a redefinition of the vocabulary of an ERDF ontology, which now becomes finite. We call the modified semantics, the *ERDF #n-stable model semantics* (for $n \in I\!N$). We define the ERDF #n-stable model semantics of O similarly to the ERDF stable model semantics of O, but now only the interpretation of the terms in $V_O^{\#n}$ is considered.

The ERDF #n-stable model semantics also extends RDFS entailment from RDF graphs to ERDF ontologies. Query answering under the ERDF #n-stable model semantics is decidable. Moreover, 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 [3] for an extended logic program $\Pi_O^{\#n}$. Finally, we would like to mention that the complexity results of Proposition 3.1 also hold for the ERDF #n-stable model semantics, after replacing (i) stable by $\#n$-stable, (ii) $\models^{st}$ by $\models^{st\#n}$, and (iii) n_O by n.

REFERENCES

[1] A. Analyti, G. Antoniou, C. V. Damásio, and G. Wagner. Extended RDF as a Semantic Foundation of Rule Markup Languages. *Journal of Artificial Intelligence Research (JAIR)*, 32:37–94, 2008.

[2] R. Berger. The Undecidability of the Dominoe Problem. *Memoirs of the American Mathematical Society*, 66:1–72, 1966.

[3] M. Gelfond and V. Lifschitz. Logic programs with Classical Negation. In *ICLP'90*, pages 579–597, 1990.

[4] P. Hayes. RDF Semantics. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`.

[5] 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.