Stable Model Theory for Extended RDF Ontologies*

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

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

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

1 Introduction

The idea of the Semantic Web is to describe the meaning of web data in a way suitable for automated reasoning. This means that descriptive data (meta-data) in machine readable form are to be stored on the web and used for reasoning. Due to its distributed and world-wide nature, the Web creates new problems for knowledge representation research. In [2], the following fundamental theoretical problems have been identified: negation and contradictions, open-world versus closed-world assumptions, and rule systems for the Semantic Web. For the time being, the first two issues have been circumvented by discarding the facilities to

* This research has been partially funded by European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (www.rewerse.net).

Y. Gil et al. (Eds.): ISWC 2005, LNCS 3729, pp. 21–36, 2005.

introduce them, namely negation and closed-world assumptions. Though the web ontology language OWL [13], which is based on description logic (DL), includes a form of classical negation through class complements, this form is limited. This is because, to achieve decidability, classes are formed based on specific class constructors and negation on properties is not considered. Rules constitute the next layer over the ontology languages of the Semantic Web and, in contrast to DL, allow arbitrary interaction of variables in the body of the rules. The widely recognized need of having rules in the Semantic Web [10,14] has restarted the discussion of the fundamentals of closed-world reasoning and the appropriate mechanisms to implement it in rule systems, such as the computational concept of *negation-as-failure*.

The RDF(S) recommendation [6] provides the basic constructs for defining web ontologies and a solid ground to discuss the above issues. RDF(S) is a special predicate logical language that is restricted to existentially quantified conjunctions of atomic formulas, involving binary predicates only. Thus, RDF(S) does not support negation and rules. In [18], it was argued that a database, as a knowledge representation system, needs two kinds of negation, namely *weak negation* $\sim$ (expressing negation-as-failure or not-truth) and *strong negation* $\neg$ (expressing explicit negative information or falsity) to be able to deal with partial information. In [19], this point was made for the Semantic Web as a framework for knowledge representation in general. In the present paper we make the same point for the Semantic Web language RDF and show how it can be extended to accommodate the two negations of partial logic [7], as well as derivation rules. We call the extended language *Extended RDF* and denote it by *ERDF*. The model-theoretic semantics of ERDF, called *ERDF stable model semantics*, is developed based on partial logic [7].

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

Partial logic allows also to distinguish between properties (similarly, classes) that are completely represented in a knowledge base and those that are not. The classification if a property is completely represented or not is up to the owner of the knowledge base: the owner must know for which properties there is complete information and for which there is not. Clearly, in the case of a completely represented (*closed*) predicate p, negation-as-failure implies falsity, and the underlying completeness assumption is also called *Closed-World Assumption (CWA)*. A CWA for p is represented in our framework through the inclusion of the derivation rule $\neg p(?x, ?y) \leftarrow \sim p(?x, ?y)$ (for a closed class c, the correspond-

ing CWA is $\neg rdf{:}type(?x, c) \leftarrow \sim rdf{:}type(?x, c)$). In the case of an incompletely represented (*open*) predicate p, negation-as-failure is not applicable and explicit negative information has to be supplied along with ordinary (positive) information. In particular, the inclusion of the derivation rule $\neg p(?x, ?y) \leftarrow \sim p(?x, ?y)$ will not affect the semantics of p. Unfortunately, neither classical logic nor Prolog supports this distinction between "closed" and "open" predicates. Classical logic supports only open-world reasoning. On the contrary, Prolog supports only closed-world reasoning, as *negation-as-failure* is the only negation mechanism supported. For arguments in favor of the combination of closed and open world reasoning in the same framework, see [1].

Specifically, in this paper:

1. We extend RDF graphs to ERDF graphs with the inclusion of strong negation, and then to ERDF ontologies (or ERDF knowledge bases) with the inclusion of general derivation rules. ERDF graphs allow to express existential positive and negative information, whereas general derivation rules allow inferences based on formulas built using the connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$ and the quantifiers $\forall$, $\exists$.
2. We extend the vocabulary of RDF(S) with the terms $erdf{:}TotalProperty$ and $erdf{:}TotalClass$, representing metaclasses of total properties and total classes, on which the open-world assumption applies.
3. We extend RDFS interpretations to ERDF interpretations including both truth and falsity extensions for properties and classes. Then, we define *coherent ERDF interpretations* by imposing coherence on all properties. In the developed model-theoretic semantics of ERDF, we consider only coherent ERDF interpretations. Thus, total properties and classes become synonymous to classical properties and classes.
4. We extend RDF graphs to ERDF formulas that are built from positive triples using the connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$ and the quantifiers $\forall$, $\exists$. Then, we define ERDF entailment between two ERDF formulas, extending RDFS entailment between RDF graphs.
5. We define the ERDF models, Herbrand interpretations, minimal Herbrand models, and stable models of ERDF ontologies. We show that stable model entailment on ERDF ontologies extends RDFS entailment on RDF graphs.
6. We show that if all properties are total, classical (boolean) Herbrand model reasoning and stable model reasoning coincide. In this case, we make an open-world assumption for all properties and classes.

The rest of the paper is organized as follows: In Section 2, we extend RDF graphs to ERDF graphs and ERDF formulas. Section 3 defines ERDF interpretations and ERDF entailment. We show that ERDF entailment extends RDFS entailment. In Section 4, we define ERDF ontologies and the Herbrand models of an ERDF ontology. In Section 5, we define the stable models of an ERDF ontology and show that stable model entailment extends RDFS entailment. Section 6 reviews related work and Section 7 concludes the paper.

2 Extending RDF Graphs with Negative Information

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

According to RDF concepts [12,6], URI references are used for naming web resources. A URI reference consists of two parts: a namespace URI ns and a local name ln, and is denoted by $ns{:}ln$. A plain literal is a string "s", where s is a sequence of Unicode characters, or a pair of a string "s" and a language tag t, denoted by "s"@t. A typed literal is a pair of a string "s" and a datatype URI reference d, denoted by "s"^^d. A (Web) *vocabulary* V is a set of URI references and/or literals (plain or typed). We denote the set of all URI references by URI, the set of all plain literals by $\mathcal{PL}$, the set of all typed literals by $\mathcal{TL}$, and the set of all literals by $\mathcal{LIT}$.

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

Below we extend the notion of RDF triple to allow for both positive and negative information.

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

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

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

element of $L(V)$. We denote the set of variables appearing in F by $Var(F)$, and the set of free variables[1] appearing in F by $FVar(F)$. □

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

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

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

Intuitively, an ERDF graph G represents an existentially quantified conjunction of $ERDF$ triples. Specifically, let $G = \{tr_1, ..., tr_n\}$ be an $ERDF$ graph, and let $Var(G) = \{x_1, ...x_k\}$. Then, G represents the formula $\exists x_1, ...x_k\ tr_1 \wedge ... \wedge tr_n$. Following the RDF terminology [12], the variables of an ERDF graph are called *blank nodes*, and intuitively denote anonymous web resources.

Note that as an RDF graph is a set of RDF triples [12,6], an RDF graph is also an ERDF graph.

3 ERDF Interpretations

In this section, we extend RDF(S) semantics by allowing for partial properties and classes. In particular, we define ERDF interpretations and satisfaction of an ERDF formula. For simplicity, we disregard RDF(S) containers, collections, and reification, as no special semantic conditions are imposed on these, and thus can be included by a straightforward extension.

Below we define a partial interpretation as an extension of a simple interpretation [6], where each property is associated not only with a truth extension but also with a falsity extension allowing for partial properties.

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

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

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

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

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

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

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

Note that as an ERDF graph represents an existentially quantified conjunction of ERDF triples, the above definition applies also to ERDF graphs. Specifically, let G be an ERDF graph representing the formula $F = \exists x_1, ...x_k\ tr_1 \wedge ... \wedge tr_n$. We say that a partial interpretation I *satisfies* the ERDF graph G ($I \models G$) iff $I \models F$.

We are now ready to define an ERDF interpretation over a vocabulary V as an extension of an RDFS interpretation [6], where each property and class is associated not only with a truth extension but also with a falsity extension, allowing for both partial properties and partial classes. Additionally, an ERDF interpretation gives special semantics to terms from the ERDF vocabulary.

The vocabulary of RDF, $\mathcal{V}_{RDF}$, and the vocabulary of RDFS, $\mathcal{V}_{RDFS}$, are defined in [6]. The *vocabulary of ERDF*, $\mathcal{V}_{ERDF}$, is a set of *URI* references in the *erdf*: namespace. Specifically, the set of ERDF predefined classes is $\mathcal{C}_{ERDF} = \{erdf{:}TotalClass,\ erdf{:}TotalProperty\}$. We define $\mathcal{V}_{ERDF} = \mathcal{C}_{ERDF}$. Intuitively, instances of the metaclass *erdf*:*TotalClass* are classes c that satisfy totalness, meaning that each resource belongs to the truth or falsity extension of

c. Similarly, instances of the metaclass $erdf{:}TotalProperty$ are properties p that satisfy totalness, meaning that each pair of resources belongs to the truth or falsity extension of p.

Definition 7 (ERDF interpretation). An *ERDF interpretation* I of a vocabulary V is a partial interpretation of $V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$, extended by the new ontological categories $Cls_I \subseteq Res_I$ for classes, $TCls_I \subseteq Cls_I$ for total classes, and $TProp_I \subseteq Prop_I$ for total properties, as well as the class-truth extension mapping $CT_I : Cls_I \to \mathcal{P}(Res_I)$, and the class-falsity extension mapping $CF_I : Cls_I \to \mathcal{P}(Res_I)$, such that:

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

Note that the semantic conditions of ERDF interpretations may impose constraints to both the truth and falsity extensions of properties and classes.

Definition 8 (Coherent ERDF interpretation). An ERDF interpretation I of a vocabulary V is *coherent* iff for all $x \in Prop_I$, $PT_I(x) \cap PF_I(x) = \emptyset$. □

Coherent ERDF interpretations enforce the constraint that a pair of resources cannot belong to both the truth and falsity extensions of a property. Since $rdf{:}type$ is a property, this constraint also implies that a resource cannot belong to both the truth and falsity extensions of a class.

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

According to RDFS semantics, the only source of RDFS-inconsistency is the appearance of an ill-typed XML literal in the RDF graph (possibly causing an *XML clash*, for details see [6]). An ERDF graph can be ERDF-inconsistent[2], not only due to the appearance of an ill-typed XML literal in the ERDF graph, but also due to the additional semantic condition for coherent ERDF interpretations.

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

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

Proposition 1. Let I be an ERDF interpretation of a vocabulary V and let $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Then,

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

Definition 9 (Classical ERDF interpretation). A (coherent) ERDF interpretation I of a vocabulary V is *classical* iff for all $x \in Prop_I$, $PT_I(x) \cup PF_I(x) = Res_I \times Res_I$. □

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

Proposition 2. Let I be an ERDF interpretation of a vocabulary V and let $V' = V \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. Then,

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

The following definition defines ERDF entailment between two ERDF formulas.

Definition 10 (ERDF Entailment). Let F, F' be ERDF formulas. We say that F *ERDF-entails* F' ($F \models^{ERDF} F'$) iff for every ERDF interpretation I, if $I \models F$ then $I \models F'$. □

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

The following proposition shows that an RDF graph is RDFS satisfiable iff it is ERDF satisfiable.

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

Proposition 3. Let G be an RDF graph such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$. Then, there is an RDFS interpretation that satisfies G iff there is an ERDF interpretation that satisfies G.

The following proposition shows that ERDF entailment extends RDFS entailment from RDF graphs to ERDF formulas.

Proposition 4. Let G, G' be RDF graphs such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$ and $V_{G'} \cap \mathcal{V}_{ERDF} = \emptyset$. Then, $G \models^{RDFS} G'$ iff $G \models^{ERDF} G'$.

4 ERDF Ontologies

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

Definition 11 (ERDF rule, ERDF program). An *ERDF rule* r over a vocabulary V is an expression of the form: $G \leftarrow F$, where $F \in L(V)$ is called *condition* and $G \in L(V|\{\neg\})$ is called *conclusion*. We assume that no bound variable in F appears free in G. We denote the set of variables and the set of free variables of r by $Var(r)$ and $FVar(r)$[3], respectively. Additionally, we write $Cond(r) = F$ and $Concl(r) = G$.
An *ERDF program* P is a set of ERDF rules over some vocabulary V. We denote the set of URI references and literals appearing in P by V_P. □

Definition 12 (ERDF ontology). An *ERDF ontology* (or *knowledge base*) is a pair $O = \langle G, P \rangle$, where G is an ERDF graph and P is an ERDF program. □

The following definition defines the models of an ERDF ontology.

Definition 13 (Satisfaction of an ERDF rule and an ERDF ontology). Let I be an ERDF interpretation of a vocabulary V.

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

Definition 14 (Skolemization of an ERDF graph). Let G be an ERDF graph. The *skolemization function* of G is an 1:1 mapping $sk_G : Var(G) \rightarrow URI$, where for each $x \in Var(G)$, $sk_G(x)$ is an artificial URI denoted by $G{:}x$. The set $sk_G(Var(G))$ is called the *Skolem vocabulary* of G.

The *skolemization* of G, denoted by $sk(G)$, is the ground ERDF graph derived from G after replacing each variable $x \in Var(G)$ by $sk_G(x)$. □

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

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

Proposition 5. Let G be an ERDF graph and let I be an ERDF interpretation. Then, $I \models sk(G)$ implies $I \models G$.

Definition 15 (Vocabulary of an ERDF ontology). Let $O = \langle G, P \rangle$ be an ERDF ontology. The *vocabulary* of O is defined as $V_O = V_{sk(G)} \cup V_P \cup \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{V}_{ERDF}$. □

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

Definition 16 (Herbrand interpretation, Herbrand model of an ERDF ontology). Let $O = \langle G, P \rangle$ be an ERDF ontology and let I be an ERDF interpretation of V_O. I is a *Herbrand interpretation* of O iff:

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

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

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

5 Minimal Herbrand Interpretations and Stable Models

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

For example, let $p, s, o \in URI$, let $G = \{p(s, o)\}$ and let $O = \langle G, \emptyset \rangle$, Then, there is a Herbrand model I of O such that $I \models p(o, s)$, whereas we want $\sim p(o, s)$ to be satisfied by all intended models of O, as p is not a total property[4] and $p(o, s)$ cannot be derived from O (negation-as-failure).

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

Definition 17 (Herbrand interpretation ordering). Let $O = \langle G, P \rangle$ be an ERDF ontology. Let $I, J \in \mathcal{I}^H(O)$. We say that J *extends* I, denoted by $I \leq J$ (or $J \geq I$), iff $Prop_I \subseteq Prop_J$, and for all $p \in Prop_I$, it holds $PT_I(p) \subseteq PT_J(p)$ and $PF_I(p) \subseteq PF_J(p)$. □

[4] On total properties and classes, the open-world assumption applies.

The intuition behind Definition 17 is that by extending a Herbrand interpretation, we extend both the truth and falsity extension for all properties, and thus (since *rdf:type* is a property), for all classes.

Definition 18 (Minimal Herbrand Interpretations). Let O be an ERDF ontology and let $\mathcal{I} \subseteq \mathcal{I}^H(O)$. We define $minimal(\mathcal{I}) = \{I \in \mathcal{I} \mid \nexists J \in \mathcal{I} : J \neq I$ and $J \leq I\}$. □

Let $I, J \in \mathcal{I}^H(O)$, we define $[I, J]_O = \{I' \in \mathcal{I}^H(O),\ I \leq I' \leq J\}$. Additionally, we define the *minimal Herbrand models* of O, as $\mathcal{M}^{min}(O) = minimal(\mathcal{M}^H(O))$.

However minimal Herbrand models do not give the intended semantics to all ERDF rules. This is because ERDF rules are derivation and not implication rules. Derivation rules are often identified with implications. For nonmonotonic rules (e.g. with negation-as-failure), this is no longer the case.

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

Definition 19 (Grounding of an ERDF program). Let V be a vocabulary and r be an ERDF rule. We denote by $[r]_V$ the set of rules that result from r if we replace each variable $x \in FVar(r)$ 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 [7] (which, on extended logic programs, are equivalent [7] to Answer Sets [5]).

Definition 20 (Stable model). Let $O = \langle G, P \rangle$ be an ERDF ontology and let $M \in \mathcal{I}^H(O)$. We say that M is a *stable model* of O iff there is a chain of Herbrand interpretations of O, $I_0 \leq ... \leq I_k$ such that $I_{k-1} = I_k = M$ and:

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

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

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

Proposition 6. Let $O = \langle G, P \rangle$ be an ERDF ontology and let $M \in \mathcal{M}^{st}(O)$. It holds $M \in \mathcal{M}^H(O)$.

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

Proposition 7. Let $O = \langle G, P \rangle$ be an ERDF ontology, such that *rdfs:subclass*(*rdf:Property*, *erdf:TotalProperty*) $\in G$. Then, $\mathcal{M}^{st}(O) = \mathcal{M}^H(O)$.

From Proposition 2, it follows that if $rdfs{:}subclass(rdf{:}Property, erdf{:}TotalProperty) \in G$ then each $M \in \mathcal{M}^H(O)$ is a classical ERDF interpretation. Therefore, the above proposition shows that classical (boolean) Herbrand model reasoning on ERDF ontologies is a special case of stable model reasoning.

Similarly to [5,8,7], stable models do not preserve Herbrand model satisfiability. For example, let $O = \langle \emptyset, P \rangle$, where $P = \{p(s,o) \leftarrow {\sim}p(s,o)\}$, and $p, s, o \in URI$. Then, $\mathcal{M}^{st}(O) = \emptyset$, whereas there is a Herbrand model of O that satisfies $p(s,o)$.

Definition 21 (Stable model entailment). Let $O = \langle G, P \rangle$ be an ERDF ontology and let F be an ERDF formula. We say that O *entails* F under the *(ERDF) stable model semantics*, denoted by $O \models^{st} F$ iff for all $M \in \mathcal{M}^{st}(O)$, $M \models F$. □

For example, let $O = \langle \emptyset, P \rangle$, where $P = \{p(s,o) \leftarrow {\sim}q(s,o)\}$ and $p, q, s, o \in URI$. Then, $O \models^{st} {\sim}q(s,o) \wedge p(s,o)$. Let $O = \langle G, P \rangle$, where $G = \{rdfs{:}subclass(rdf{:}Property, erdf{:}TotalProperty)\}$ and P is as in the previous example. Then, $O \models^{st} q(s,o) \vee p(s,o)$, but $O \not\models^{st} {\sim}q(s,o)$ and $O \not\models^{st} p(s,o)$. This is the desirable result, since q is a total property, and thus in contrast to the previous example, an open-world assumption is made for q. As another example, let $p, s, o \in URI$, let $G = \{p(s,o)\}$, and let $P = \{\neg p(?x, ?y) \leftarrow {\sim}p(?x, ?y)\}$. Then, $\langle G, P \rangle \models^{st} {\sim}p(o,s) \wedge \neg p(o,s)$ (note that P contains a CWA on p). Let $G = \{rdf{:}type(p, erdf{:}TotalProperty),\ p(s,o)\}$ and let P be as in the previous example. Then, $\langle G, P \rangle \models^{st} \forall ?x\ \forall ?y\ \ (p(?x, ?y) \vee \neg p(?x, ?y))$ (see Proposition 1), but $\langle G, P \rangle \not\models^{st} {\sim}p(o,s)$ and $\langle G, P \rangle \not\models^{st} \neg p(o,s)$. Indeed, the CWA in P does not affect the semantics of p, since p is a total property.

Let us now see a more involved example[5]. Consider the following ERDF program P, specifying some rules for concluding that a country is not a member state of the European Union (EU).

$$(r_1) \quad \neg rdf{:}type(?x, EUMember) \leftarrow rdf{:}type(?x, AmericanCountry).$$
$$(r_2) \quad \neg rdf{:}type(?x, EUMember) \leftarrow rdf{:}type(?x, EuropeanCountry), {\sim}rdf{:}type(?x, EUMember).$$

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

$\neg rdf{:}type(Russia, EUMember).$ $\quad$ $rdf{:}type(Canada, AmericanCountry).$
$rdf{:}type(Austria, EUMember).$ $\quad$ $rdf{:}type(Italy, EuropeanCountry).$
$rdf{:}type(?x, EuropeanCountry).$ $\quad$ $\neg rdf{:}type(?x, EUMember).$

Using stable model entailment on O, it can be concluded that Austria is a member of EU, that Russia and Canada are not members of EU, and that it exists a European Country which is not a member of EU. However, it is also concluded that Italy is not a member of EU, which is a wrong statement. This is because G does not contain complete information of the European countries

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

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

The following proposition shows that stable model entailment extends RDFS entailment from RDF graphs to ERDF ontologies.

Proposition 8. Let G, G' be RDF graphs such that $V_G \cap \mathcal{V}_{ERDF} = \emptyset$, $V_{G'} \cap \mathcal{V}_{ERDF} = \emptyset$, and $V_{G'} \cap sk_G(Var(G)) = \emptyset$. It holds: $G \models^{RDFS} G'$ iff $< G, \emptyset > \models^{st} G'$.

Below we define the stable answers of a query F w.r.t. an ERDF ontology.

Definition 22 (Stable answers). Let $O = \langle G, P \rangle$ be an ERDF ontology. A *query* F is an ERDF formula. The *(ERDF) stable answers* of F w.r.t. O are defined as follows: $Ans^{st}_O(F) = \{v : FVar(F) \rightarrow V_O \mid \forall M \in \mathcal{M}^{st}(O) : M \models v(F)\}$, where $v(F)$ is the formula F after replacing all the free variables x in F by $v(x)$. □

An ERDF ontology $O = \langle G, P \rangle$ is called *simple* if each rule in P has the form $L_0 \leftarrow L_1, ..., L_k, \sim L_{k+1}, ..., \sim L_n$, where each L_i is an ERDF triple (positive or negative). The following proposition shows that the stable answers of a query F w.r.t. a simple ERDF ontology can be computed through Answer Set Programming [5] on an extended logic program (ELP).

Proposition 9. Let $O = \langle G, P \rangle$ be a simple ERDF ontology and let F be an ERDF formula. We can define an extended logic program Π_O and a corresponding formula F' such that: The answers of F' according to the answer set semantics [5] of Π_O coincide with $Ans^{st}_O(F)$.

Intuitively, Π_O is generated as follows: (i) each $[\sim|\neg]p(s, o) \in L(V_O|\{\sim, \neg\})$ is represented by $[\sim|\neg]Holds(s, p, o)$, where $Holds$ is a conventional predicate name and p becomes a term, (ii) $sk(G)$ is represented as a set of facts, and (iii) semantics implicit in the definition of an ERDF interpretation is represented as rules. Π_O is the union of the rules generated in (ii-iii).

6 Related Work

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

TRIPLE [15] is a rule language for the Semantic Web supporting RDF and a subset of OWL Lite [13]. It is based on F-Logic [11]. Part of the semantics of the RDF(S) vocabulary is represented in the form of pre-defined rules and not

as semantic conditions on interpretations. TRIPLE includes a form of negation-as-failure under the well-founded semantics [4]. Strong negation is not used.

Flora-2 [20] is a rule-based object-oriented knowledge base system for reasoning with semantic information on the Web. It is based on F-logic [11] and supports metaprogramming, nonmonotonic multiple inheritance, logical database updates, encapsulation, dynamic modules, and two kinds of weak negation (specifically, Prolog negation and well-founded negation [4]). In Flora-2, anonymous resources are handled through skolemization (similarly to our theory).

Notation 3 (N3) provides a more human readable syntax for RDF and also extends RDF by adding numerous pre-defined constructs ("built-ins") for being able to express rules conveniently (see [17]). Remarkably, N3 contains a built-in (log:definitiveDocument) for making restricted completeness asumptions and another built-in (log:notIncludes) for expressing simple negation-as-failure tests. The addition of these constructs was motivated by use cases. However, N3 does not have any direct formal semantics for these constructs, and does not provide strong negation. In an extended version of this paper we will show how these N3 constructs can be mapped to ERDF.

OWL-DL [13] is an ontology representation language for the Semantic Web, that is a syntactic variant of the $\mathcal{SHOIN}(\mathbf{D})$ description logic and a decidable fragment of first-order logic. However, the need for extending the expressive power of OWL-DL with rules has initiated several studies, including the SWRL (Semantic Web Rule Language) proposal [10]. In [9], it is shown that this extension is in general undecidable. For an overview of (decidable) approaches of combining Description Logics with rules, see [3]. In several of these approaches, entailment on the extended with rules DL is based on first-order logic, that is both the DL component and the logic program are viewed as a set of first-order logic statements. Thus, negation-as-failure, closed-world-assumptions, and nonmonotonic reasoning cannot be supported. In contrast in our work, we support both weak and strong negation, and allow closed-world and open-world reasoning on a selective basis.

7 Conclusions

In this paper, we extended RDF graphs to ERDF graphs by allowing negative triples, and then to ERDF ontologies with the inclusion of derivation rules, allowing freely appearance of (meta)properties and (meta)classes in the body and head of the rules, all logical factors $\sim$, $\neg$, $\forall$, $\exists$, $\supset$, $\wedge$, $\vee$ in the body of the rules, and strong negation $\neg$ in the head of the rules. Moreover, the RDF(S) vocabulary was extended with the terms *erdf*:*TotalProperty* and *erdf*:*TotalClass*. We have developed the model-theoretic semantics of ERDF ontologies, called *ERDF stable model semantics*, showing that stable model entailment extends RDFS entailment on RDF graphs. We have shown that classical (boolean) Herbrand model reasoning is a special case of our semantics, when all properties are total. In this case, similarly to classical logic, an open-world assumption is made for all properties and classes. Allowing totalness of properties and classes to

be declared on a selective basis and the explicit representation of closed-world assumptions (as derivation rules) enables the combination of open-world and closed-world reasoning in the same framework. For simple ERDF ontologies, our semantics can be computed through Answer Set Programming [5]. Future work concerns the support of datatype maps, including XSD datatypes, and the extension of the ERDF vocabulary to other useful ontological categories possibly in accordance with [16].

References

1. A. Analyti, G. Antoniou, C. V. Damasio, and G. Wagner. Negation and Negative Information in the W3C Resource Description Framework. *Annals of Mathematics, Computing & Teleinformatics (AMCT)*, 1(2):25–34, 2004.
2. Tim Berners-Lee. Design issues - architectual and philosophical points. Personal notes, 1998. Available at `http://www.w3.org/DesignIssues/`.
3. E. Franconi and S. Tessaris. Rules and Queries with Ontologies: A Unified Logical Framework. In *Second International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2004)*, pages 50–60, 2004.
4. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
5. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th International Conference on Logic Programming*, pages 579–597. MIT Press, 1990.
6. Patrick Hayes. RDF Semantics. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`.
7. 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.
8. H. Herre and G. Wagner. Stable Models are Generated by a Stable Chain. *Journal of Logic Programming*, 30(2):165–177, 1997.
9. I. Horrocks and P. F. Patel-Schneider. A Proposal for an OWL Rules Language. In *13th International Conference on World Wide Web (WWW'04)*, pages 723–731. ACM Press, 2004.
10. 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/`.
11. M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, 1995.
12. 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/`.
13. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/2004/REC-owl-features-20040210/`.
14. The rule markup initiative (ruleml). Available at `http://www.ruleml.org`.
15. M. Sintek and S. Decker. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *First International Semantic Web Conference on The Semantic Web (ISWC2002)*, pages 364–378. Springer-Verlag, 2002.

16. H. J. ter Horst. Extending the RDFS Entailment Lemma. In *3rd International Semantic Web Conference (ISWC2004)*, pages 77–91, 2004.
17. Tim-Berners-Lee. Notation 3 - An RDF language for the Semantic Web. W3C Recommendation, 1998. Available at `http://www.w3.org/DesignIssues/Notation3.html`.
18. G. Wagner. A Database Needs Two Kinds of Negation. In *3rd Symposium on Mathematical Fundamentals of Database and Knowledge Base Systems (MFDBS'91)*, pages 357–371. Springer-Verlag, 1991.
19. G. Wagner. Web Rules Need Two Kinds of Negation. In *1st International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR'03)*, pages 33–50. Springer-Verlag, December 2003.
20. Guizhen Yang and Michael Kifer. Inheritance and Rules in Object-Oriented Semantic Web Languages. In *2nd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RULEML'03)*, pages 95–110, 2003.