Embeddings of Simple Modular Extended RDF

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

[1] CENTRIA, Departamento de Informática da Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal. cd@di.fct.unl.pt
[2] Institute of Computer Science, FORTH-ICS, Crete, Greece. analyti@ics.forth.gr
[3] Institute of Computer Science, FORTH-ICS, and Department of Computer Science, University of Crete, Crete, Greece. antoniou@ics.forth.gr

Abstract. The Extended Resource Description Framework has been proposed to equip RDF graphs with weak and strong negation, as well as derivation rules, increasing the expressiveness of ordinary RDF graphs. In parallel, the Modular Web framework enables collaborative and controlled reasoning in the Semantic Web. In this paper we exploit the use of the Modular Web framework to specify the modular semantics for Extended Resource Description Framework ontologies.

1 Introduction

The Extended Resource Description Framework [5] (ERDF) provides a model theoretical semantics for RDF graphs allowing negative triples, and ontologies defined by first-order rules including the two forms of negation, weak and strong. However, no means of combining different ontologies is specified. The necessity of mechanisms to encapsulate and organize knowledge in the Semantic Web is essential [13, 6, 8, 10], and the ERDF framework has been extended to allow the specification of import and export declarations of classes and properties, resulting in the Modular ERDF framework [3]. The semantics of the modular ERDF framework has also been defined model theoretically, but it was lacking a declarative rule-based semantics for implementing the system.

In parallel, the Modular Web Framework (MWeb) is a proposal to address the issues of programming-in-the-wide faced by the new Semantic Web rule-engines [2, 4]. MWeb defines general constructs to allow sharing of knowledge in the Semantic Web provided by logic based knowledge bases, including scoped open and closed world assumptions with contextualized and global interpretation of predicates. The MWeb framework is constructed, compatible and based on Rule Interchange Format (RIF) guidelines fostering immediate integration with RDF [12]. MWeb provides two semantics designated MWebWFS and MWebAS with a solid theory based on the two major semantics of extended logic programming, respectively, Well-Founded Semantics with Explicit Negation [1] and Answer Sets [9]. A compiler of MWeb into XSB Prolog is available[4] making use of the tabling features to guarantee termination of recursive rules with negation. It provides separate interface and implementation of rulebases with modular and independent compilation.

The major contribution of the paper is the specification of the semantics of ERDF reasoning entirely in the MWeb framework, including alignment with RIF, support of

[4] The system can be downloaded at http://centria.di.fct.unl.pt/~cd/mweb/

RDF and RDFS entailment, as well extensions to the original ERDF semantics for dealing with closed classes and properties. These results complement the mapping of simple modular ERDF ontologies into MWeb rulebases defined in [7]. Thus reasoning on simple modular ERDF ontologies can be achieved through our MWeb implementation[5], and in particular supporting modular reasoning over RDF(S) ontologies.

The paper is organized as follows. In Section 2 we illustrate how simple modular ERDF ontologies are mapped into MWeb rulebases. Next Section 3, specifies the support of ERDF reasoning by MWeb logic rules instead of the formal model-theoretical presentation of [3]. The paper finishes with some conclusions.

2 The MWeb Embedding of ERDF Ontologies

Simple modular ERDF ontologies [3] allow the combination of knowledge in different ontologies. Specifically, a simple modular ERDF ontology (SMEO) is a set of simple `r`-ERDF ontologies. The language of simple `r`-ERDF ontologies allows the use of ordinary triples s`.[`p `->>` o`]` and negated triples `neg` s`.[`p `->>` o`]` in the ERDF graph, where s, p and o are respectively the subject, predicate and object of the statement. Additionally, it allows to construct programs using deductive rules to derive new (extended) triples by rules having bodies formed by combining the connectives `naf` (weak negation), `neg` (strong negation), and conjunction. Moreover, provides mechanisms to define modules of knowledge, which are described by an interface and formed by an ERDF graph and a program. Finally, it provides a means to query other rulebases via qualified literals of the form $Lit@URI$ in rules. Details can be found in [3, 7].

The MWeb framework requires for each rulebase (module of knowledge) the definition of an *interface* document and of the corresponding *rulebase* (*logic*) document. The MWeb interface is formed by a sequence of declarations. First, the name of the rulebase is stated via a `rulebase` declaration followed by an IRI. Optional base IRI and prefixes can be declared for simplifying writing of classes and property names, via a `base` and `prefix` declarations. Other interfaces may be recursively included via a special `import` declaration. This mechanism will be used to import the interfaces declaring the classes and properties defined by RIF, RDF, RDFS and ERDF. An optional `vocabulary` declaration can be used to list the vocabulary of the rulebase. Next, follow two blocks of declarations. The first block defines the predicates being defined in the MWeb rulebase, and correspond to a generalization of export declarations found in logic programming based languages. The second block correspond to generalization of import declarations. The interesting feature of the MWeb framework is that besides scope (i.e. internal, local, or global), different reasoning modes can be associated to predicates (i.e. definite, open, closed, or normal). This allows control of monotonicity of reasoning by the producer and consumer of the knowledge. In this work, all properties and classes are defined global (meaning that it can be defined in multiple rulebases) and normal (meaning that weak negation can be used). The semantics of all MWeb constructs can be found in [4] as well as additional motivation. In [7] it is defined the translation of simple modular ERDF ontologies into the MWeb framework, whose general interface document is illustrated in Figure 1.

[5] The MWeb system is available at `http://centria.di.fct.unl.pt/~cd/mweb`.

```
:- rulebase 'Nam_O'.
:- import('erdf.mw',interface).
:- vocabulary rdf:'_1',..., rdf:'_n'.
:- defines global normal class(mw:Vocabulary).
% For each class c exported to r_1,... r_n
:- defines global normal class(c) visible to 'r_1',...,'r_n'.
% For each property p exported to r_1,... r_n
:- defines global normal property(p) visible to 'r_1',...,'r_n'.
% For each class c imported from s_1,... s_m
:- uses normal class(c)from 's_1',...,'s_m'.
% For each property p imported from s_1,... s_m
:- uses normal property(p)from 's_1',...,'s_m'.
% Let u_1,...,u_d be the r-ERDF ontologies on which O depends
:- uses normal class(mw:Vocabulary)from'u_1',...,'u_d'.
```

Fig. 1. Simple Modular ERDF Ontologies Interface in MWeb

The first declaration in Figure 1 identifies the rulebase, while the `import` directive includes in the interface the necessary declarations for supporting ERDF reasoning, namely the vocabularies of RDF, RDFS and ERDF. The `erdf.mw` interface and corresponding rulebase will be presented later on, and implements in MWeb itself the underlying semantics of modular ERDF ontologies, including RIF, RDFS and RDF combination. The next declaration lists a limited number of container membership properties to be included in the vocabulary, in the case that at least one occurs in the graph or in the program. However, the property `rdf:_1` is always declared by the `erdf` ontology. The vocabulary of the rulebase is collected in the pre-defined class `mw:Vocabulary` of the MWeb framework and made visible to the allowed potential importing rulebases. The rulebase vocabulary is used for providing the domain for (scoped) negation as failure, open and closed world assumptions.

The next group declares the exported classes and properties, via the `defines` declaration of MWeb. The important point is that all classes and properties are defined normal and global. This means that all classes and properties exported can be used and redefined (`global`), and that rules can use weak negation (`normal`) and thus are non-monotonic. The visibility list states where the class or property can be used.

The subsequent `uses` declarations correspond to the import part of the interface, and are used in normal mode (weak negation allowed). The meaning of the importing list is obvious. The last `use` declaration extends the vocabulary of the rulebase with the vocabulary of the rulebases in the dependencies (directly or indirectly used modules). Notice that the vocabulary used in the current interface and corresponding program documents is automatically included, and need not to be declared.

The translation of the logic document of an `r`-*ERDF ontology* is immediate since the syntax used to represent rules in ERDF and MWeb is almost identical. In general, the MWeb logic document can start with an optional `import` declaration which allows textual inclusion of the rules found in the imported document. Afterwards, the fact and rules can be stated. The exact translation of simple modular ERDF programs can be

found in [7], which is not presented for lack of space. Several examples of concrete and full MWeb logic documents will appear in the rest of the paper.

3 Semantics of Modular ERDF Ontologies in MWeb

In this section we specify the semantics of ERDF entailment through MWeb rulebases. This will be achieved incrementally and hierarchically, by providing first the definition of the used RIF primitive predicates. Afterwards, the semantics of RDF will be defined and made compatible with RIF as prescribed in [12], and subsequently a rulebase will define RDFS. Finally, we take care of the features of ERDF entailment.

3.1 RIF Support

The supported Rule Interchange Format dialect implements fully the semantics of membership and subclass, frames, and equality (partially). Regarding connectives, the usual binary conjunction, as well as strong and weak negations are supported. In order to maintain compatibility with RIF and generality, all properties and classes are assumed to be global and normal, allowing for the use of weak negation in the bodies, and thus monotonicity cannot be guaranteed. The MWeb syntax recognizes frames of the form `?O.[?A1->>?V1,...,?An->>?Vn]` which internally are translated into a conjunction `'->'(?A1,?O,?V1),...,'->'(?An,?O,?Vn)` of the ternary predicate `'->'/3`. The other binary RIF predicates '=' (equality), '#' (member), and '##' (subclass) have the exact syntax of RIF, and for ease of presentation are infix operators. The semantics of these predicates are provided by the rulebase of Figure 2.

RIF interface (rif.mw)

```
:- rulebase 'http://www.w3.org/2007/rif'.
:- prefix rif='http://www.w3.org/2007/rif#'.
:- defines internal normal name:'='/2, name:'#'/2, name:'##'/2, name:'->'/3.
```

RIF rulebase (rif.rb)

```
% RIF member relation
?O # ?CL :- ?O # ?SCL, ?SCL ## ?CL.
neg ?O # ?SCL :- neg ?O # ?CL, ?SCL ## ?CL.
% RIF subclass relation
?C1 ## ?C3 :- ?C1 ## ?C2, ?C2 ## ?C3.
% RIF equality theory.
?T = ?T :- ?T # mw:Vocabulary.
?T1 = ?T2 :- ?T2 = ?T1.
?T1 = ?T3 :- ?T1 = ?T2, ?T2 = ?T3.
neg ?T1 = ?T2 :- neg ?T2 = ?T1.
neg ?T1 = ?T3 :- ?T1 = ?T2, neg ?T2 = ?T3.
% RIF frames obtained by equality reasoning.
?O.[?P ->> ?V] :- ?O1.[?P ->> ?V], ?O = ?O1.
?O.[?P ->> ?V] :- ?O.[?P1 ->> ?V], ?P = ?P1.
?O.[?P ->> ?V] :- ?O.[?P ->> ?V1], ?V = ?V1.
```

Fig. 2. MWeb Rulebase Implementing RIF Relations

The interface document just defines the RIF primitive predicates being implemented using predicate indicators (name/arity), and these are hidden with the internal keyword.

Notice the use of the special prefix `name` which is associated with the empty string. Class inheritance is captured by the first rule in document '`rif.rb`'. Mark the need to have a rule for taking care of the case where it is known that something does not belong to the extension of some class (second rule). For subclass relation it is only required transitivity, which does not have a dual negative rule.

Equality rules implement reflexivity, commutativity and transitivity. Reflexivity is applied to the declared vocabulary collected in the pre-defined class `mw:Vocabulary`. The equality rules are restricted to frames, and cannot handle complex terms[6]. Notice that this is very similar to the rules of `owl:sameAs` in the OWL2 RL profile [11].

3.2 RDF Semantics

The semantics of the combination of RDF with rules is the one adopted by RIF and specified in [12]. An ordinary triple `s p o` is syntactically represented by the RIF frame `s.[p ->> o]`. RIF lists are not supported because would require introducing complex terms in the language.

The support of RDF entailment is immediate and can be found in Figure 3. All the classes and properties of the RDF vocabulary are declared in the interface document '`rdf.mw`'. Notice also the declaration of the prefix `rdf` to simplify writing of URIs using Compact URI notation (CURIEs). A class declaration `class(`$CURIE$`)` is short for `?_ #` $CURIE$, while the `property(`$CURIE$`)` is syntactic sugar for the RIF frame?_.[$CURIE$->>?_] resulting in better looking interface documents. The `?_` occurrences represent anonymous variables.

By the recommendation governing RIF-RDF compatibility, the predicates '`#`'/2 and `rdf:type` should be made equivalent; this is achieved by the first rules in document '`rif.rb`'. The only rule necessary for RDF entailment states that any predicate of a triple must have type `rdf:Property`. Then, the axiomatic RDF triples are listed, concluding with the special treatment of RDF container membership properties.

The RDF container membership properties are handled by external calls to Prolog underlying system, since they are infinitely many (`rdf:_1`, `rdf:_2`, etc...), and their full inclusion would result in the generation of an infinite number of answers for some non-ground queries. The rule only fires if the subject of the triple is bound at query time with a ground atom.

3.3 RDFS Semantics

RDF Schema entailment is more complex to specify, but immediate. According to RIF-RDF compatibility every RIF subclass instance is also an `rdfs:subClassOf` instance (but not vice-versa). Afterwards, all the RDF schema inference rules and axiomatic triples are encoded; container membership properties are treated as in the implementation of RDF. Besides XML and plain literals, no other datatypes are handled.

A snippet of the MWeb implementation of RDFS entailment is present in Figure 4, and adopts the complete RDFS entailment rules of [14]. The interface is very similar to the one of RDF, adapted to the corresponding vocabulary.

[6] Full equality is not necessary for ERDF, whose terms are just URIs and literals.

RDF interface (rdf.mw)

```
:- rulebase  'http://www.w3.org/1999/02/22-rdf-syntax-ns'.
:- prefix  rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'.
:- import( 'rif.mw', interface ).

:- vocabulary rdf:nil, rdf:'_1'.
:- defines internal normal class(rdf:Property), ...
:- defines internal normal property(rdf:type), ...
```

RDF rulebase (rdf.rb)

```
:- import( 'rif.rb', rulebase ).
% RDF compatibility with RIF makes # and rdf:type equivalent
 ?X # ?Y :- ?X.[ rdf:type ->> ?Y].
 ?X.[ rdf:type ->> ?Y] :- ?X # ?Y.
% RDF Entailment rule
 ?Z.[ rdf:type ->> rdf:Property] :- ?_.[?Z ->> ?_].

% RDF Axiomatic triples
 rdf:type.[rdf:type->>rdf:Property].        rdf:subject.[rdf:type->>rdf:Property].
 rdf:predicate.[rdf:type->>rdf:Property].  rdf:object.[rdf:type ->> rdf:Property].
 rdf:first.[rdf:type ->> rdf:Property].      rdf:rest.[rdf:type ->> rdf:Property].
 rdf:value.[rdf:type ->> rdf:Property].           rdf:nil.[rdf:type ->> rdf:List].

% Infinitely many membership properties. Uses side-effects to restrict.
?X.[rdf:type ->> rdf:Property] :-
 External(name:atom(?X),prolog), External(name:atom_concat(rdf:'_',?N,?X ),prolog),
 External(name:is_number_atom(?N),prolog).
```

Fig. 3. MWeb Rulebase Implementing RDF Entailment

3.4 ERDF Semantics

The Extended Resource Description Framework vocabulary introduces the notions of total and closed class, as well as total and closed property, and a mechanism to express complementary properties, whose interface file '`erdf.mw`' is depicted below:

```
:- rulebase 'http://erdf.org'.
:- prefix erdf='http://erdf.org#'.
:- import( 'rdfs.mw', interface ).

:- defines internal normal class(erdf:TotalClass),
   class(erdf:PositivelyClosedClass), class(erdf:NegativelyClosedClass).

:- defines internal normal class(erdf:TotalProperty),
   class(erdf:PositivelyClosedProperty), class(erdf:NegativelyClosedProperty).

:- defines internal normal property(erdf:complementOf).
```

Totalness is enforced by declaring the class and property having `erdf:TotalClass` and `erdf:TotalProperty` type, respectively, corresponding to open world assumptions with respect to the declared vocabulary. Closed classes can be declared using type `erdf:PositivelyClosedClass` or `erdf:NegativelyClosedClass`, similarly `erdf:PositivelyClosedProperty`, and `erdf:NegativelyClosedProperty` can be used to declare closed properties. These correspond to closed world assumptions with respect to the declared vocabulary.

The semantics of the ERDF constructs is specified in the rulebase '`erdf.rb`'. Obviously, the document '`erdf.rb`' starts by importing the defining rules of RDFS, RDF and RIF rulebases with the initial declaration `:- import('rdfs.rb', rulebase).`

RDFS interface (rdfs.mw)

```
:- rulebase 'http://www.w3.org/2000/01/rdf-schema'.
:- prefix rdfs='http://www.w3.org/2000/01/rdf-schema#'.
:- import( 'rdf.mw', interface ).

:- defines internal normal class(rdfs:Resource), class(rdfs:Literal),
   class(rdfs:Datatype), class(rdfs:Class), ...

:- defines internal normal property(rdfs:domain), property(rdfs:range),
   property(rdfs:subClassOf), property(rdfs:subPropertyOf), ...
```

RDFS rulebase (rdfs.rb)

```
:- import( 'rdf.rb', rulebase ).

% RDFS compatibility into RIF requires including ## into rdfs:subClassOf
?X.[rdfs:subClassOf ->> ?Y] :- ?X ## ?Y.

% Some of RDFS entailment rules
?Z.[ rdf:type ->> ?Y] :- ?X.[rdfs:domain ->> ?Y], ?Z.[?X ->> ?W].
?W.[ rdf:type ->> ?Y] :- ?X.[rdfs:range ->> ?Y], ?Z.[?X ->> ?W].
?Z.[ rdf:type ->> ?Y] :- ?X.[rdfs:subClassOf ->> ?Y], ?Z.[rdf:type ->> ?X].

?X.[rdfs:subClassOf ->> ?X] :- ?X.[rdf:type ->> rdfs:Class] .
?X.[rdfs:subClassOf ->> ?Z] :-
   ?X.[rdfs:subClassOf ->> ?Y], ?Y.[rdfs:subClassOf ->> ?Z].
   .
   .
   .
% other entailment and RDFS axiomatic triples follow
```

Fig. 4. MWeb Rulebase Implementing RDFS Entailment

The relationship to RIF primitive predicates is extended to negative extensions of the predicates by the following rules:

```
neg ?X # ?Y :- neg ?X.[ rdf:type ->> ?Y].
neg ?X.[rdf:type ->> ?Y] :- neg ?X # ?Y.
neg ?X.[rdfs:subClassOf ->> ?Y] :- neg ?X ## ?Y.
```

The next rule extends RDF entailment by assigning the type `rdf:Property` to properties which are used in negative triples:

```
?Z.[rdf:type->>rdf:Property] :- neg ?_.[?Z->>?_].
```

ERDF extends RDFS with rules for propagating "downwards" in the hierarchy negative class and negative property extensions:

```
neg ?Z.[rdf:type ->> ?X] :- ?X.[rdfs:subClassOf->>?Y], neg ?Z.[rdf:type->>?Y].
neg ?Z1.[?X ->> ?Z2] :- ?X.[rdfs:subPropertyOf->>?Y], neg ?Z1.[?Y->>?Z2].
```

The remaining rules take care of specificities of ERDF entailment itself. The semantics of total classes and properties are captured by the next rules encoding open-world assumptions. Since these rules require the use of negation as failure, we have to guarantee grounding of the free variables in order to avoid unsoundness of reasoning. The grounding of variables is made with respect to the rulebase declared vocabulary, and therefore is a scoped negation as failure:

```
neg ?Z.[rdf:type->>?X] :- ?X.[rdf:type ->> erdf:TotalClass],
   ?Z#mw:Vocabulary, naf ?Z.[rdf:type->>?X].
?Z.[rdf:type->>?X] :- ?X.[rdf:type ->> erdf:TotalClass],
   ?Z#mw:Vocabulary, naf neg ?Z.[rdf:type->>?X].
```

```
neg ?Z1.[?X->>?Z2] :- ?X.[rdf:type ->> erdf:TotalProperty],
    ?Z1#mw:Vocabulary, ?Z2#mw:Vocabulary, naf ?Z1.[?X->>?Z2].
?Z1.[?X->>?Z2] :- ?X.[rdf:type ->> erdf:TotalProperty],
    ?Z1#mw:Vocabulary, ?Z2#mw:Vocabulary, naf neg ?Z1.[?X->>?Z2].
```

The semantics of closed classes and properties are captured by using one of the rules in the definition for total ones. For instance, a positively closed class means that its positive instances are exhaustive, therefore all the remaining individuals in the vocabulary are known to not belonging to the class (this is captured in the first rule below). Negatively closed classes have a dual interpretation. The notion of positively and negatively closed properties are similar.

```
neg ?Z.[rdf:type->>?X] :- ?X.[rdf:type->>erdf:PositivelyClosedClass],
    ?Z#mw:Vocabulary, naf ?Z.[rdf:type->>?X].
?Z.[rdf:type->>?X] :- ?X.[rdf:type->>erdf:NegativelyClosedClass],
    ?Z#mw:Vocabulary, naf neg ?Z.[rdf:type->>?X].
neg ?Z1.[?X->>?Z2] :- ?X.[rdf:type->>erdf:PositivelyClosedProperty],
    ?Z1#mw:Vocabulary, ?Z2#mw:Vocabulary, naf ?Z1.[?X->>?Z2].
?Z1.[?X->>?Z2] :- ?X.[rdf:type->>erdf:NegativelyClosedProperty],
    ?Z1#mw:Vocabulary, ?Z2#mw:Vocabulary, naf neg ?Z1.[?X->>?Z2].
```

For legacy applications to be able to express negative triples in ordinary RDF graphs, the ERDF vocabulary includes a mechanism to state that properties are complementary with the property `erdf:complementOf`. The net effect is the exchange of the positive and negative instances of the complementary properties:

```
neg ?S.[?P->>?O] :- ?P.[erdf:complementOf->>?Q], ?S.[?Q->>?O].
neg ?S.[?P->>?O] :- ?Q.[erdf:complementOf->>?P], ?S.[?Q->>?O].
?S.[?P->>?O] :- ?P.[erdf:complementOf->>?Q], neg ?S.[?Q->>?O].
?S.[?P->>?O] :- ?Q.[erdf:complementOf->>?P], neg ?S.[?Q->>?O].
```

Finally, the axiomatic triples of ERDF are included which basically state that all classes in the ERDF vocabulary are subclasses of `rdfs:Class` and that the special property `erdf:complementOf` has domain and range `rdf:Property`. For lack of space these are not included here, but are trivial to state.

A direct translation into extended logic programming of the MWeb rulebases has been defined in [7], which uses a quad representation `'->'('m',p,s,o)` predicate to state that triple `s p o` is true at MWeb rulebase m. Briefly, a uses `class(`C`)` declaration in the interface of m generates, for each rulebase 'r_i' in its from list, the rule `'#'('m',?X,`C`) :- '#'('`r_i`',?X,`C`)`. A uses `property(`P`)` generates, for each rulebase 'r_i' in its from list, the rule `'->'('m',`P`,?S,?O):-'->'('`r_i`',`P`,?S,?O)`. Program rules are translated by introducing an extra first argument 'm' to all literals in the rule, except for qualified literals $L@o$ whose new (first) argument is 'o'. This translation has been shown sound and complete with respect to the simple modular ERDF ontologies semantics [3], and can be used for query answering of simple modular ERDF ontologies.

4 Discussion and Conclusions

This work provides a rule-based declarative specification of ERDF entailment in simple modular ERDF ontologies, via the embedding of simple modular ERDF ontologies into the MWeb framework filling in the details in [7]. The specification is based on a novel

program transformation. It reports, to the best of our knowledge, the first complete approach combining for the first time RIF and RDFS semantics, and extends it to the case of graphs capable of expressing negative information and ontologies with open, closed world assumptions, and scoped negation as failure. The representation power is complemented with features for combining modularly ontologies in the Semantic Web.

A complete working system resorting to the MWeb implementation in XSB Prolog 3.2 has been developed, and is available for download with promising results. We performed a first comparison using the W3C's Wine ontology, determing CPU times for RDFS inference without equality reasoning (discarding loading and compile times). Briefly, Jena's (2.6.2) inbuilt RDFSReasoner was 2 times slower than our MWeb implementation, while Jena's Generic reasoner was 100 times slower. Euler Yap (Eye 3414) shown to be 4 times slower and CWM-1.2.1 was 100 times slower.

References

1. J. J. Alferes, C. V. Damásio, and L. M. Pereira. A Logic Programming System for Nonmonotonic Reasoning. *Journal of Automated Reasoning*, 14(1):93–147, 1995.
2. A. Analyti, G. Antoniou, and C. V. Damásio. A Principled Framework for Modular Web Rule Bases and Its Semantics. In *Proc. of KR-2008*, pages 390–400. AAAI press, 2008.
3. A. Analyti, G. Antoniou, and C. V. Damásio. A Formal Theory for Modular ERDF Ontologies. In *Web Reasoning and Rule Systems (RR 2009)*, volume 5837 of *LNCS*, pages 212–226. Springer, 2009.
4. A. Analyti, G. Antoniou, and C. V. Damásio. MWeb: a Principled Framework for Modular Web Rule Bases and its Semantics. Accepted in ACM Transactions on Computational Logic (TOCL), 2010.
5. 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*, 32:37–94, 2008.
6. J. Bao, G. Voutsadakis, G. Slutzki, and V. Honavar. Package-based description logics. In *Modular Ontologies*, volume 5445 of *LNCS*, pages 349–371. Springer, 2009.
7. C. V. Damásio, A. Analyti, and G. Antoniou. Implementing Simple Modular ERDF ontologies. In *Proc. of 19th European Conference on Artificial Intelligence*, 2010. To appear.
8. F. Ensan. Formalizing Ontology Modularization through the Notion of Interfaces. In *16th Int. Conf. on Knowledge Engineering: Practice and Patterns (EKAW-2008)*, pages 74–82, 2008.
9. M. Gelfond and V. Lifschitz. Logic programs with Classical Negation. In *7th International Conference on Logic Programming (ICLP'90)*, pages 579–597, 1990.
10. B. C. Grau, B. Parsia, and E. Sirin. Ontology integration using epsilon-connections. In *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, pages 293–320. Springer, 2009.
11. OWL 2 RL in RIF W3C Working Group Note 22 June 2010. Edited by Dave Reynolds. Latest version available at `http://www.w3.org/TR/rif-owl-rl/`.
12. RIF RDF and OWL Compatibility W3C Recommendation 22 June 2010. Edited by Jos de Bruijn. Latest version available at `http://www.w3.org/TR/rif-rdf-owl/`.
13. L. Serafini, A. Borgida, and A. Tamilin. Aspects of Distributed and Modular Ontology Reasoning. In *19th Int. Joint Conf. on Artificial Intelligence*, pages 570–575, 2005.
14. 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.