Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 271 (2011) 3–22
www.elsevier.com/locate/entcs

A Prolog-based Query Language for OWL

Jesús M. Almendros-Jiménez [1,2]

Dpto. de Lenguajes y Computación
Universidad de Almería
04120-Spain

Abstract

In this paper we investigate how to use logic programming (in particular, Prolog) as query language against OWL resources. Our query language will be able to retrieve data and meta-data about a given OWL based ontology. With this aim, firstly, we study how to define a query language based on a fragment of Description Logic, then we show how to encode the defined query language into Prolog by means of logic rules and finally, we identify Prolog goals which correspond to queries.

Keywords: OWL, Logic Programming, Semantic Web

1 Introduction

The *Semantic Web* framework [11,13] proposes that Web *data* represented by *HMTL* and *XML* have to be enriched by means of *meta-data*, in which modeling is mainly achieved by means of the *Resource Description Framework (RDF)* [22] and the *Web Ontology Language (OWL)* [16]. RDF and OWL are proposals of the *W3C consortium* [3] for ontology modeling. OWL is syntactically layered on RDF whose underlying model is based on triples. The *RDF Schema (RDFS)* [10] is also a *W3C* proposal and enriches RDF with specific vocabularies for meta-data. RDFS/RDF and OWL can be used for expressing both data and meta-data. OWL can be considered as an extension of RDFS

[1] The author work has been partially supported by the Spanish MICINN under grant TIN2008-06622-C03-03.
[2] Email: jalmen@ual.es
[3] http://www.w3.org.

doi:10.1016/j.entcs.2011.02.008

in which a richer vocabulary allows to express new relationships. OWL offers more complex relationships than RDF(S) between entities including means to limit the properties of classes with respect to the number and type, means to infer that items with various properties are members of a particular class and a well-defined model of property inheritance. OWL is based on the so-called *Description Logic (DL)* [8], which is a family of logics (i.e. *fragments*) with different expressivity power. Most of fragments of Description Logic are subsets or variants of $\mathcal{C}^2$, the subset of *first-order logic (FOL)* extended with counting quantifiers, with formulas without function symbols and maximum two variables, which is known to be decidable [15]. Description Logic can therefore also be understood as an attempt to address the major drawbacks of using FOL for knowledge representation and inference, and also the syntax of DL allows a variable-free notation. The most prominent fragment of DL is $\mathcal{SROIQ}$ which is the basis of the new standarized OWL 2. OWL 2 semantics has been defined in [26], in which a *direct semantics* is defined based on Description Logic, and in [27] a *RDF-based semantics* is provided.

In this paper we investigate how to use logic programming (in particular, Prolog) as query language against OWL resources. Our query language will be able to retrieve data and meta-data about a given OWL based ontology. With this aim, firstly, we study how to define a query language based on a fragment of Description Logic, then we show how to encode the defined query language into Prolog by means of logic rules and finally, we identify Prolog goals which correspond to queries.

Basically, our work goes towards the use of logic programming as query language for the Semantic Web. It follows our previous research line about the use of logic programming for the handling of Web data. In particular, we have studied the encoding in logic programming of the XML query language *XPath* in [2,1], and the encoding in logic programming of the XML query language *XQuery* in [3,5], studying extensions of *XQuery* for the handling of RDF and OWL in [4,7,6]. In this framework, we would like to study how OWL querying and reasoning can be achieved by means of logic programming in order to be integrated with the proposal of the implementation of *XQuery* in logic programming.

1.1 Description Logic

Description Logic is a formalism for expressing relationships between *concepts*, between *roles*, between *concepts* and *roles*, and between *concepts*, *roles* and *individuals*. Formulas of Description Logic can be used for representing knowledge, that is, concept descriptions, about a domain of interest. Typically, Description Logic is used for representing a **TBox** (*terminological box*)

and the **ABox** (*assertional box*). The **TBox** describes concept (and role) *hierarchies* (i.e., relations between concepts and roles) while the **ABox** contains relations between individuals, concepts and roles. Therefore we can see the **TBox** as the meta-data description, and the **ABox** as the description about data.

In this context, we can distinguish between (1) *reasoning tasks* and (2) *querying tasks* from a given ontology. In both cases, a certain *inference procedure* should be present in order to deduce new relationships from a given ontology. The most typical (1) *reasoning tasks*, with regard to a given ontology, include: (a) *instance checking*, that is, whether a particular individual is a member of a given concept, (b) *relation checking*, that is, whether two individuals hold a given role, (c) *subsumption*, that is, whether a concept is a subset of another concept, (d) *concept consistency*, that is, consistency of the concept relationships, and (e) a more general case of consistency checking is *ontology consistency* in which the problem is to decide whether a given ontology has a model. However, one can be also interested in (2) *querying tasks* such as: (a) *instance retrieval*, which means to retrieve all the individuals of a given concept entailed by the ontology, and (b) *property fillers retrieval* which means to retrieve all the individuals which are related to a given individual with respect to a given role.

OWL/DL reasoning is a topic of research of increasing interest in the literature. Most of DL reasoners (for instance, *Racer* [20], FaCT++ [30], Pellet [28]) are based on tableaux based decision procedures. Based on logic programming, the *DLog* system [23] reasons with an ontology by means of the encoding into Prolog and the use of the PTTP theorem prover. The *KAON2* tool [9,19] is based on the encoding of the $\mathcal{SHIQ}$ fragment into disjuntive *Datalog* [18]. They have studied a resolution-based inference.

We believe that an interesting research line would be to design a *query language* whose aim is to express such reasoning and querying tasks. In other words, the study of some kind of formalism in which we can express the kind of task (i.e. reasoning and querying task) one wants to achieve with respect to a given ontology. Such a language should be equipped with some kind of *formulas* for representing the query. In addition, such a query language should be equipped with an *inference mechanism* in order to reason with the ontology. Such inference mechanism could be based on an *entailment* relationship. The query language we propose will be based on Description Logic formulas which can contain *free* variables. Free variables represent the elements of the formula for which we want to retrieve values. Such values should represent the names of *concepts*, *roles* and *individuals* satisfying a given query (i.e. the DL formula). In such a case, we would obtain the *answers* to a given querying task. In the

case of formulas without free variables, the query would represent a reasoning task, and the answer would be true or false.

The definition of a set of entailment rules for RDFS and OWL has attracted the attention of the Semantic Web community. An entailment relationship defines which relationships can be deduced from a given ontology. In this context, the authors of [25] have observed that the rules of entailment of the official RDF Semantics specification are not complete, and have suggested for the case of RDFS, to identify a fragment which encompasses the essential features of RDFS, which preserves the original semantics, be easy to formalize and can serve to prove results about its properties. With this aim they have defined a fragment of RDFS that covers the crucial vocabulary of RDFS, they have proved that it preserves the original RDF semantics, and avoids vocabulary and axiomatic information that only serves to reason about the structure of the language itself and not about the data it describes. The studied fragment of RDFS lifts the structural information into the semantics of the language hiding them from developers and users. They have given a sound and entailment relationship for a fragment of RDF including `rdf:type`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain` and `rdfs:range`. In the case of OWL, there is a proposal for extending rules for entailment with RDFS to OWL. The so-called pD^* approach [29] is a proposal for an extension of the RDFS vocabulary with some elements of OWL: `owl:FunctionalProperty`, `owl:InverseFunctionalProperty`, `owl:sameAs`, `owl:SymmetricProperty`, `owl : TransitiveProperty` and `owl:inverseOf`. For such a fragment, they have defined a complete set of simple entailment rules. The pD^* approach has been successfully applied to the *SAOR (Scalable Authoritative OWL reasoner)* [17], a system which focuses on a good performance with RDF and OWL data.

In this line, our approach considers a set of entailment rules for a fragment of OWL. Such fragment differs from the fragment of the pD^* [29], neither pD^* is include in our fragment, nor our fragment is included in pD^*, but our fragment includes the RDFS fragment of [25]. In addition, our fragment of DL allows to encode the entailment relationship by means of logic programming, in particular, by Prolog. For this reason, we have studied the relationship between Description Logic and Logic Programming.

1.2 Description Logic and Logic Programming

In this area of research, some authors [14,31] have studied the intersection of Description Logic and Logic Programming, in other words, which fragment of Description Logic can be expressed in Logic Programming. In [14], they have defined the so-called *Description Logic Programming (DLP)*, which cor-

responds with a fragment of $\mathcal{SHIQ}$. With this aim, firstly, the fragment of DL is encoded into a fragment of FOL; and after the fragment of FOL is encoded into logic programming. Other fragments of OWL/DL can be also encoded into logic programming, in particular, Volz [31] has encoded fragments of $\mathcal{SHOIN}$ into *Datalog*, *Datalog(=)*, *Datalog(=,IC)* and *Prolog(=,IC)*; where "=" means "with equality", and "IC" means "with Integrity constraints". Some other proposals have encoded the fragment $\mathcal{SHIQ}$ into *disjunctive Datalog* [18], and into *Datalog(IC,≠,not)* [12], where "not" means "with negation".

In our proposal, we have focused on one of the Volz's fragments which can be encoded into *Datalog*. It is a fragment of $\mathcal{SHOIN}$, which includes the OWL elements: `rdf:type`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, `owl : equivalentProperty`, `owl:equivalentClass`, `rdfs:domain`, `rdfs:range`, `owl: someValuesFrom`, `owl:hasValue` and `owl:allValuesFrom`, and handles `owl:union` and `owl:intersection` operators, and `owl:TransitiveProperty`, `owl:SymmetricProperty` and `owl:inverseOf` properties. All of them are used with some restrictions. We believe that our proposal could be extended to other fragments of $\mathcal{SHOIN}$ studied in [31].

Our work can be intended as an extension of the DLP framework in the following sense. The encoding of DL into logic programming defines an entailment relationship based on the rules. However, our encoding differs from DLP encoding: instead of encoding class and role names as Prolog predicates, we encode them as *Prolog atoms*. In fact, our encoding only uses a predicate called `triple` which defines by means of *Prolog facts* the relationships (i.e. the **ABox** and **TBox** elements) of the ontology. Therefore the ontology can be easily stored in secondary memory with efficient access. As a consequence of such encoding concept and role names are now *first-class citizens*, and they can be handled as individuals in the corresponding Prolog program and goals. In addition, in our approach, and in contrast with the DLP approach, *complex concepts* in Description Logic are handled by means of *Prolog terms*. For instance, $\forall P.C$ is represented as a Prolog term `forall(P,C)`. It also allows to handle complex concepts also as first-class citizens. Our work can be also considered as an extension of DLP framework because we investigate a more flexible *query language* than in DLP. By studying the entailment relationship between Description Logic formulas of the given fragment, we are able to provide semantics to a more flexible query language than the underlying query language in the DLP framework. For instance, we are able to entail in our framework typing formulas of the style $\exists P.C(I)$, that is, I is an individual of type $\exists P.C$, in the spite of $\exists P.C(I)$ is not explicitely declared in the **ABox**. But more interesting, such a typing (i.e., $\exists P.C$) can be retrieved as a answer of our query language. It is not possible with the DLP encoding because $\exists P.C$

formulas can not be handled as first-class citizens. Moreover, P and C can work as query result, providing more flexible queries. In other words, our DLP extension can be considered as *second order extension* (i.e. predicate names in DLP can be replaced by variables) of the DLP approach. For such extension, we provide an entailment relationship. The entailment relationship defined in our proposal is also an extension of the underlying entailment relationship of the DLP fragment. Our entailment relationship is able to infer new DL formulas of the kind $\exists P.C \sqsubseteq D$ when they are not explicitely declared in the **TBox**. Such new DL formulas would correspond with the inference of *new rules* from the DLP encoding of the selected fragment. In summary, our entailment relationship can obtain the same statements about the **ABox**: individual assertions and property fillers than the DLP approach for the same fragment, but we can also obtain new statements about the **TBox**: subclass and subproperty assertions with respect to the DLP approach.

We have developed a prototype of our approach which can be downloaded from `http://indalog.ual.es/OProlog`. We have tested our prototype with several examples of ontologies including the running example presented bellow. With respect to the implementation we have to make the following remarks:

- We have implemented the OWL-based query language using the SWI-Prolog interpreter.
- We have used the RDFS library of SWI-Prolog [32] for implementing loading of OWL triples. The RDFS library of SWI-Prolog has limited querying capabilities. Our proposal can be considered as an extension of such library for OWL querying.
- The syntactic structure of entailment rules makes that the implementation in Prolog loops: the predicate `triple` can call to itself with the same arguments. However, we have solved that problem by implementing an small Prolog interpreter which runs Prolog rules in such a way that it memorizes the facts about the predicate `triple`, and avoids calls with the same arguments.

The structure of the paper is as follows. Section 2 will present the fragment of DL of our proposal. Section 3 will define the query language proposed from the fragment. Section 4 will describe the encoding of the query language in Prolog. Finally, Section 5 will conclude and present future work.

2 Web Ontology Language

In this section we will show what kind of ontologies will be allowed in our framework. It will also define the entailment relationship.

Fig. 1. An Example of Ontology

TBox	
(1) $Man \sqsubseteq Person$	*(2)* $Woman \sqsubseteq Person$
(3) $Person \sqcap \exists author\_of.Manuscript \sqsubseteq Writer$	*(4)* $Paper \sqcup Book \sqsubseteq Manuscript$
(5) $Book \sqcap \exists topic.\{\text{“XML”}\} \sqsubseteq XMLbook$	*(6)* $Manuscript \sqcap \exists reviewed\_by.Person \sqsubseteq Reviewed$
(7) $Manuscript \sqsubseteq \forall rating.Score$	*(8)* $Manuscript \sqsubseteq \forall topic.Topic$
(9) $author\_of \equiv writes$	*(10)* $average\_rating \sqsubseteq rating$
(11) $authored\_by \equiv author\_of^{-}$	*(12)* $\top \sqsubseteq \forall\ author\_of.Manuscript$
(13) $\top \sqsubseteq \forall\ author\_of^{-}.Person$	*(14)* $\top \sqsubseteq \forall\ reviewed\_by.Person$
(15) $\top \sqsubseteq \forall\ reviewed\_by^{-}.Manuscript$	
ABox	
(1) Man(“Abiteboul”)	
(2) Man(“Buneman”)	*(3) Man(“Suciu”)*
(4) Book(“Data on the Web”)	*(5) Book(“XML in Scottland”)*
(6) Paper(“Growing XQuery”)	*(7) Person(“Anonymous”)*
(8) author_of(“Abiteboul”,“Data on the Web”)	*(9) authored_by(“Data on the Web”, “Buneman”)*
(10) author_of(“Suciu”,“Data on the Web”)	*(11) author_of(“Buneman”, “XML in Scottland”)*
(12) writes(“Simeon”,“Growing XQuery”)	*(13) reviewed_by(“Data on the Web”, “Anonymous”)*
(14) reviewed_by(“Growing XQuery”,“Almendros”)	*(15) average_rating(“Data on the Web”, “good”)*
(16) rating(“XML in Scottland”,“excellent”)	*(17) average_rating(“Growing XQuery”, “good”)*
(18) topic(“Data on the Web”,“XML”)	*(19) topic(“Data on the Web”,“Web”)*
(20) topic(“XML in Scottland”,“XML”)	

An ontology $\mathcal{O}$ in our framework contains a **TBox** including a sequence of definitions $\mathcal{T}_1, \ldots, \mathcal{T}_n$ of the form given in Table 1 below, where E, F are *class (i.e. concept) descriptions* of equivalence type (denoted by $E, F \in \mathcal{E}$) of the form:

$$E ::= C_0 \mid E_1 \sqcap E_2 \mid \exists P.\{O\}$$

In addition, C is a *class (i.e. concept) description of left-hand side type* (denoted by $C \in \mathcal{L}$), of the form:

$$C ::= C_0 \mid C_1 \sqcap C_2 \mid \exists P.\{O\} \mid C_1 \sqcup C_2 \mid \exists P.C$$

$\mathcal{T} ::=$	
$C \sqsubseteq D \mid$	(`rdfs:subClassof`)
$E \equiv F \mid$	(`owl:equivalentClass`)
$P \sqsubseteq Q \mid$	(`rdfs:subPropertyOf`)
$P \equiv Q \mid$	(`owl:equivalentProperty`)
$P \equiv Q^- \mid$	(`owl:inverseOf`)
$P \equiv P^- \mid$	(`owl:SymmetricProperty`)
$P^+ \sqsubseteq P \mid$	(`owl:TransitiveProperty`)
$\top \sqsubseteq \forall P^-.D \mid$	(`rdfs:domain`)
$\top \sqsubseteq \forall P.D$	(`rdfs:range`)

Table 1
Definitions for an ontology $\mathcal{O}$

and D is a *class (i.e. concept) description of right-hand side type* (denoted by $D \in \mathcal{R}$), of the form:

$$D ::= A \mid D_1 \sqcap D_2 \mid \exists P.\{O\} \mid \forall P.D$$

In all previous cases, C_0 is an *atomic class* (i.e. *class name*), P, Q are *property (i.e. role) names* and O is an *individual name*. In addition, the **ABox** contains a sequence of definitions $\mathcal{A}_1, \ldots, \mathcal{A}_m$ of the form:

$$\mathcal{A} := P(A, B) \mid C_0(A) \mid \top(A)$$

where P is a *property name*, C_0 is a *class name*, and A, B are *individual names*.

Basically, the proposed subset of DL restricts the form of class descriptions in right and left hand sides of subclass and class equivalence definitions, and in individual assertions. Such restriction is required according to [31] in order to be able to encode the fragment of DL into logic programming. Following [31], the universal quantification is only allowed in the right hand sides of DL formulas, which corresponds in the encoding to the occurrence of the same quantifier in the left hand sides (i.e. heads) of rules. The existential quantification only occurs in the right hand by the same reason. Union formulas are required to occur in the left hand sides of DL formulas, which corresponds in the encoding to the definition of two rules.

Table 2
Inference Calculus $\vdash_{OI}$

Rule Name	Inference	
(**Eq1**)		$\vdash_{OI} E \equiv E$
(**Eq2**)	$E \equiv F$, $F \equiv G$	$\vdash_{OI} E \equiv G$
(**Eq3**)	$E \equiv F$	$\vdash_{OI} F \equiv E$
(**Eq4**)	$E \sqsubseteq F$, $F \sqsubseteq E$	$\vdash_{OI} E \equiv F$
(**Sub1**)	$E \equiv F$	$\vdash_{OI} E \sqsubseteq F$
(**Sub2**)	$C \sqsubseteq D$, $D \sqsubseteq E$	$\vdash_{OI} C \sqsubseteq E$
(**Sub3**)	$C \sqcup D \sqsubseteq E$	$\vdash_{OI} C \sqsubseteq E$
(**Sub4**)	$E \sqsubseteq C \sqcap D$	$\vdash_{OI} E \sqsubseteq C$
(**Sub5**)	$C_1 \sqcap C_2 \sqsubseteq D$, $E \sqsubseteq C_1$	$\vdash_{OI} E \sqcap C_2 \sqsubseteq D$
(**Sub6**)	$C_1 \sqcup C_2 \sqsubseteq D$, $E \sqsubseteq C_1$	$\vdash_{OI} E \sqcup C_2 \sqsubseteq D$
(**Sub7**)	$C \sqsubseteq D_1 \sqcap D_2$, $D_1 \sqsubseteq E$	$\vdash_{OI} C \sqsubseteq E \sqcap D_2$
(**Sub8**)	$\exists P.\{O\} \sqsubseteq D$, $Q \sqsubseteq P$	$\vdash_{OI} \exists Q.\{O\} \sqsubseteq D$
(**Sub9**)	$\exists P.C \sqsubseteq D$, $Q \sqsubseteq P$	$\vdash_{OI} \exists Q.C \sqsubseteq D$
(**Sub10**)	$\exists P.C \sqsubseteq D$, $E \sqsubseteq C$	$\vdash_{OI} \exists P.E \sqsubseteq D$
(**Sub11**)	$C \sqsubseteq \exists P.\{O\}$, $P \sqsubseteq Q$	$\vdash_{OI} C \sqsubseteq \exists Q.\{O\}$
(**Sub12**)	$C \sqsubseteq \forall P.D$, $Q \sqsubseteq P$	$\vdash_{OI} C \sqsubseteq \forall Q.D$
(**Sub13**)	$C \sqsubseteq \forall P.D$, $D \sqsubseteq E$	$\vdash_{OI} C \sqsubseteq \forall P.E$
(**Type1**)	$C(A)$, $C \sqsubseteq D$	$\vdash_{OI} D(A)$
(**Type2**)	$P(A,O)$, $\exists P.\{O\} \sqsubseteq D$	$\vdash_{OI} D(A)$
(**Type3**)	$P(A,B), C(B)$ $\exists P.C \sqsubseteq D$	$\vdash_{OI} D(A)$
(**Type4**)	$C(A)$, $P(A,B)$, $C \sqsubseteq \forall P.D$	$\vdash_{OI} D(B)$
(**Type5**)	$C(A)$, $P(B,A)$, $C \sqsubseteq \forall P^{-}.D$	$\vdash_{OI} D(B)$
(**Type6**)	$Cond(E_i)$, $\sqcap_{1 \leq i \leq n} E_i \sqsubseteq D$	$\vdash_{OI} D(A)$
(**Type7**)		$\vdash_{OI} \top(A)$
(**Prop1**)		$\vdash_{OI} P \equiv P$
(**Prop2**)	$P \equiv Q$, $Q \equiv R$	$\vdash_{OI} P \equiv R$
(**Prop3**)	$P \equiv Q$	$\vdash_{OI} Q \equiv P$
(**Prop4**)	$P \sqsubseteq Q$, $Q \sqsubseteq P$	$\vdash_{OI} P \equiv Q$
(**Prop5**)	$P \equiv Q$	$\vdash_{OI} P \sqsubseteq Q$
(**Prop6**)	$P \sqsubseteq Q$, $Q \sqsubseteq R$	$\vdash_{OI} P \sqsubseteq R$
(**Prop7**)	$P \sqsubseteq Q$, $P(A,B)$	$\vdash_{OI} Q(A,B)$
(**Prop8**)	$P \sqsubseteq Q^{-}$, $P(A,B)$	$\vdash_{OI} Q(B,A)$
(**Prop9**)	$Q^{-} \sqsubseteq P$, $Q(A,B)$	$\vdash_{OI} P(B,A)$
(**Prop10**)	$P^{+} \sqsubseteq P$, $P(A,B)$, $P(B,C)$	$\vdash_{OI} P(A,C)$
(**Prop11**)	$C(A)$, $C \sqsubseteq \exists P.\{O\}$	$\vdash_{OI} P(A,O)$

Let us see an example of an ontology $\mathcal{O}_0$ (see Figure 1). The ontology $\mathcal{O}_0$ describes *meta-data* in the **TBox** defining that the elements of *Man* and the elements of *Woman* are elements of *Person* (cases (1) and (2)); and the elements of *Paper* and elements of *Book* are elements of *Manuscript* (case (4)). In addition, a *Writer* is a *Person* who is the *author_ of* a *Manuscript* (case (3)), and the class *Reviewed* contains the elements of *Manuscript reviewed_ by* a *Person* (case (6)). Moreover, the *XMLBook* class contains the elements of *Manuscript* which have as *topic* the value "XML" ((5)). The classes *Score* and *Topic* contain, respectively, the values of the properties *rating* and *topic* associated to *Manuscript* (cases (7) and (8)). The property *average_ rating* is a subproperty of *rating* (case (10)). The property *writes* is equivalent to *author_ of* (case (9)), and *authored_ by* is the inverse property of *author_ of* (case (11)). Finally, the property *author_ of*, and conversively, *reviewed_ by*, has as domain a *Person* and as range a *Manuscript* (cases (12)-(15)).

The **ABox** describes *data* about two elements of *Book*: "Data on the Web" and "XML in Scottland" and a *Paper*: "Growing XQuery". It describes the *author_ of* and *authored_ by* relationships for the elements of *Book* and the *writes* relation for the elements of *Paper*. In addition, the elements of *Book* and *Paper* have been reviewed and rated, and they are described by means of a topic.

2.1 Entailment Relationship

Now, we would like to show an inference calculus, denoted by $\vdash_{OI}$, which defines the entailment relationship between formulas of the selected fragment. The inference calculus is shown in Table 2. In the rule **(Eq1)** E is a class name, and D is a class name in rules **(Type1)** to **(Type5)**. In the rule **(Type6)** A is an individual name, and in the rule **(Prop1)** P is a property name. In addition, in the rule **(Type6)** we have that $Cond(E) = E(A)$ if E is atomic; $Cond(E) = P(A, O)$ if $E = \exists P.\{O\}$, $Cond(E) = P(A, B), C(B)$ if $E = \exists P.C$. Finally, in the rules **(Prop1)** to **(Prop6)**, and **(Sub12)**, **(Sub13)**, P and Q can have the form S, S^- and S^+. The rules from **(Eq1)** to **(Eq4)** handle about equivalence. **(Eq1)** infers equivalence by reflexivity, **(Eq2)** infers equivalence by transitivity, and **(Eq3)** infers equivalence by symmetry. **(Eq4)** infers equivalence from the subclass relationship. The rules from **(Sub1)** to **(Sub13)** handle inference about subclasses. Cases from **(Sub3)** to **(Sub7)** define new subclass relationships from union and intersection operators. However, the calculus does not introduce new union and intersection operators. For instance $C \sqcap E \sqsubseteq D$ is not entailed from $C \sqsubseteq D$. The same can be said for $C \sqcup E \sqsubseteq D$ which is not entailed from $C \sqsubseteq D$ and $E \sqsubseteq D$. The same happens with $C \sqsubseteq D \sqcap E$ from $C \sqsubseteq D$ and $C \sqsubseteq E$. Cases from **(Sub8)**

to **(Sub13)** define new subclass relationships for complex formulas. Such entailment of complex formulas is the main contribution of our inference calculus with respect to DLP framework and the pD^* approach. In the former case, the entailment relationship can only entail formulas about individual and property fillers assertions, and not about complex formulas. In the later case, pD^* does not handle `owl:someValuesFrom`, `owl:allValuesFrom` and `owl:hasValue` and therefore such rules have not sense in pD^*. However, the pD^* handles for instance `owl:FunctionalProperty` which is not consider in our framework.

The rules **(Type1)** to **(Type7)** infer type relationships from subclass and equivalence relationships. The most relevant ones are the cases from **(Type2)** to **(Type5)** defining the meaning of complex formulas w.r.t. individuals. Finally, the rules **(Prop1)** to **(Prop11)** infer relationships about roles. The most relevant ones are the case **(Prop8)** and **(Prop9)** about the inverse of a property and the case **(Prop10)** about the transitivity relationship. Our inference calculus is able to infer new information from a given ontology. For instance, $\mathcal{O}_0 \vdash_{OI} Reviewed(''Data\ on\ the\ Web'')$, using the following **TBox** and **ABox** information of $\mathcal{O}_0$:

$$Book(''Data\ on\ the\ Web'').$$

$$Book \sqsubseteq Manuscript.$$

$$Person(''Anonymous'').$$

$$reviewed\_by(''Data\ on\ the\ Web'',''Anonymous'').$$

$$Manuscript \sqcap \exists Reviewed\_by.Person \sqsubseteq Reviewed.$$

by means of the following reasoning:

$$\vdash_{(\mathbf{Type1})} Manuscript(''Data\ on\ the\ Web'')$$

$$\vdash_{(\mathbf{Type3})} Reviewed(''Data\ on\ the\ Web'')$$

Our inference calculus can be used for proving a given Description Logic formula of the selected fragment from an ontology of the same fragment. Our inference calculus can be used for inferring all the entailments from a given ontology. The idea is to apply the rules up to a *fix point* is reached. In addition, we have designed the inference calculus in order to be implemented in logic programming, in particular, in Prolog. It forces to limit the inference capabilities of our system. The inference calculus only handles the user-defined DL complex formulas (i.e. those included in the **TBox**). For instance, we

cannot infer *new* relations like $C \sqsubseteq \forall P.D$ because it requires to check all the relations between individuals for P in the **ABox**. The same can be said for $\exists P.C \sqsubseteq D$, and $P \sqsubseteq Q$.

3 A Query Language based on Description Logic

In this section we will define the query language based on Description Logic. Such query language will introduce variables in DL formulas in order to express the values to be retrieved in the query result. In addition, our query language can handle conjunctions of DL formulas. We will use variable names starting with lower-case letters to distinguish them from non-variables.

Assuming a set $\mathcal{V}_c$ of variables for concepts $c, d, \ldots$ and a set $\mathcal{V}_p$ of variables for roles $p, q, \ldots$, and a set $\mathcal{V}_i$ of variables for individuals $a, b \ldots$, a query $\mathcal{Q}$ against of an ontology $\mathcal{O}$ is a conjunction $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$ where each $\mathcal{Q}_i$ has the form:

$\mathcal{Q} ::= C \sqsubseteq D \mid E \equiv F \mid R \sqsubseteq T \mid R \equiv T \mid P(A,B) \mid C_0(A)$

where $C \in \mathcal{L}^{\mathcal{V}}$, $D \in \mathcal{R}^{\mathcal{V}}$, $E, F \in \mathcal{E}^{\mathcal{V}}$, $R, T = P, P^-, P^+$ and $P \in \mathcal{P}^{\mathcal{V}}$ and $A, B \in \mathcal{I}^{\mathcal{V}}$, $C_0 \in \mathcal{V}_c$ or is a class name. In addition, $\mathcal{E}^{\mathcal{V}}$ contains the set of formulas of the form:

c	$\mid c \in \mathcal{V}_c$
C_0	$\mid$ atomic class
$E_1 \sqcap E_2$	$\mid E_1, E_2 \in \mathcal{E}^{\mathcal{V}}$
$\exists P.\{O\}$	$\mid P \in \mathcal{P}^{\mathcal{V}}, O \in \mathcal{I}^{\mathcal{V}}$

$\mathcal{L}^{\mathcal{V}}$ contains the set of formulas of the form:

c	$\mid c \in \mathcal{V}_c$
C_0	$\mid$ atomic class
$C_1 \sqcap C_2$	$\mid C_1, C_2 \in \mathcal{L}^{\mathcal{V}}$
$\exists P.\{O\}$	$\mid P \in \mathcal{P}^{\mathcal{V}}, O \in \mathcal{I}_{\mathcal{V}}$
$C_1 \sqcup C_2$	$\mid C_1, C_2 \in \mathcal{L}^{\mathcal{V}}$
$\exists P.C$	$\mid C \in \mathcal{L}^{\mathcal{V}}, P \in \mathcal{P}^{\mathcal{V}}$

$\mathcal{R}^{\mathcal{V}}$ contains the set of formulas of the form:

c	$\mid c \in \mathcal{V}_c$
C_0	$\mid$ atomic class
$D_1 \sqcap D_2$	$\mid D_1, D_2 \in \mathcal{R}^{\mathcal{V}}$
$\exists P.\{O\}$	$\mid P \in \mathcal{P}^{\mathcal{V}}$, $O \in \mathcal{I}_{\mathcal{V}}$
$\forall P.D$	$\mid D \in \mathcal{R}^{\mathcal{V}}$, $P \in \mathcal{P}^{\mathcal{V}}$

and finally, $\mathcal{P}^{\mathcal{V}}$ contains property names and elements of $\mathcal{V}_p$, and $\mathcal{I}^{\mathcal{V}}$ contains individual names and elements of $\mathcal{V}_i$.

As in the case of the data and meta-data definition language, the query language is restricted to a fragment of DL in order to be encoded in logic programming (i.e. in Prolog). Assuming that variable names start with lower case letters, queries are formulas like:

- *type*("*Growing XQuery*") whose meaning is "Find the type of *Growing XQuery*".
- *Person*(*p*), *Reviewed_by* (*"Growing XQuery",p)*, whose meaning is "Retrieve the reviewers of *"Growing XQuery"*" and
- *Manuscript(m), Reviewed_by(m,"Almendros")* whose meaning is "Retrieve the manuscripts in which *"Almendros"* is a reviewer"
- *Author_of*(*author*, '*Data on the Web*") whose means is "Retrieve the authors of *"Data on the Web"*". Let us remark that in this case, subproperties of *"Author_of"* are taken into account (for instance, *"Authored_by"* and *"Writes"*).

Let us remark that the previous queries can be answered from the **TBox** and the **ABox** of the ontology. Meta-data can be retrieved by means of our query language, however, queries about meta-data can be only answered from the **TBox** of the ontology. For instance, using union and intersection operators we can retrieve:

- $intersection \equiv Book \sqcap Reviewed$ whose meaning is "Find the intersections of *Book* and *Reviewed*".
- $\top \sqsubseteq \forall Writes.range$ whose meaning is "Find the ranges of *Writes*".
- $\top \sqsubseteq \forall p.Book$ whose meaning is "Find the properties whose range is *Book*".
- $\exists p.Person \sqsubseteq Book$ whose meaning is "Find the properties about *Person*'s in which the range belongs to *Book*".
- $class \sqsubseteq \exists p.\{\text{"}XML\text{"}\}$ whose meaning is "Find the classes having a role related to "XML"".

An answer of a query $\mathcal{Q}$ w.r.t. an ontology $\mathcal{O}$ and vocabulary V is a mapping θ from $DVar(\mathcal{Q})$ into V such that $\mathcal{O} \vdash_{OI} \theta(\mathcal{Q})$, where $DVar(\mathcal{Q})$ denotes the set of variables of $\mathcal{Q}$.

Finally, we have to remark that some syntactic sugar can be considered in our query language. For instance $\forall P.C(A)$ represents $a(A), a \sqsubseteq \forall P.C$, and $\exists P^-.\{O\}(A)$ represents $a \sqsubseteq \exists q.\{O\}, q \sqsubseteq P^-, a(A)$.

4 Encoding into Prolog

Now, we would like to show how to use Prolog in our framework. The role of Prolog is double. Firstly, we can encode any given ontology instance of the considered fragment into Prolog. Secondly, our inference system $\vdash_{OI}$ can be encoded into Prolog by means of rules, in such a way that a certain class of Prolog goals, which implement the Description Logic based query language can be used as query language.

4.1 Ontology Instance Encoding

The encoding of an ontology instance consists of Prolog facts of a predicate called *triple*, representing the RDF-triple based representation of OWL. In the case of the **TBox**: (a) $en(C \sqsubseteq D) = triple(en(C), rdfs : subClassOf, en(D))$; (b) $en(E \equiv F) = triple(en(E), owl : equivalentClass, en(F))$; (c) $en(P \sqsubseteq Q) = triple(en(P), rdfs : subPropertyOf, en(D))$; and (d) $en(P \equiv Q) = triple(en(P), owl : equivalentProperty, en(Q))$. In addition, $en(C)$, $en(D)$, $en(E)$, $en(F)$, $en(P)$ and $en(Q)$ represents the encoding of classes and properties in which class and property names $C, P, \ldots$ are translated as Prolog atoms $c, d, \ldots$. The special case of $\top$ is encoded as $en(\top) = owl : thing$. In addition, Prolog terms are used for representing complex DL formulas as follows: (a) $en(P^+) = trans(en(P))$; (b) $en(P^-) = inv(en(P))$; (c) $en(\forall P.C) = forall(en(P), en(C))$; (d) $en(\exists P.C) = exists(en(P), en(C))$; (e) $en(\exists P.\{O\}) = hasvalue(en(P), en(O))$; (f) $en(\sqcap_{1 \leq i \leq n} C_i) = inter([en(C_1), \ldots, en(C_n)])$ and (h) $en(\sqcup_{1 \leq i \leq n} C_i) = union([en(D_1), \ldots, en(D_n)])$. Finally, the elements of the **ABox** are also encoded as Prolog facts relating pairs of individuals by means of properties, and defining memberships to classes: (a) $en(P(A, B)) = triple(A, en(P), B)$ and (b) $en(C_0(A)) = triple(A, rdf : type, C_0)$.

4.2 Encoding of the Inference Calculus $\vdash_{OI}$

Now, the second element of the encoding consists of Prolog rules for encoding the $\vdash_{OI}$ inference calculus. The set of rules can be found in Tables 3, 4 and 5

Table 3
Encoding of the Inference Calculus I

Rule Name	Prolog Rules
(**Eq1**)	$triple(E, owl : equivalentClass, E) : -class(E).$
(**Eq2**)	$triple(E, owl : equivalentClass, G) : -triple(E, owl : equivalentClass, F),$ $triple(F, owl : equivalentClass, G).$
(**Eq3**)	$triple(E, owl : equivalentClass, F) : -triple(F, owl : equivalentClass, E).$
(**Eq4**)	$triple(E, owl : equivalentClass, F) : -triple(E, rdfs : subClassOf, F),$ $triple(F, rdfs : subClassOf, E).$
(**Sub1**)	$triple(E, rdfs : subClassOf, F) : -triple(E, owl : equivalentClass, F).$
(**Sub2**)	$triple(C, rdfs : subClassOf, E) : -triple(C, rdfs : subClassOf, D),$ $triple(D, rdfs : subClassOf, E).$
(**Sub3**)	$triple(D, rdfs : subClassOf, E) : -triple(union(U), rdfs : subClassOf, E), member(D, U).$
(**Sub4**)	$triple(E, rdfs : subClassOf, C) : -triple(E, rdfs : subClassOf, inter(I)), member(C, I).$
(**Sub5**)	$triple(inter(I2), rdfs : subClassOf, D) : -triple(inter(I1), rdfs : subClassOf, D),$ $member(C, I1), triple(E, rdfs : subClassOf, C),$ $replace(I1, C, E, I2).$
(**Sub6**)	$triple(union(U2), rdfs : subClassOf, D) : -triple(union(U1), rdfs : subClassOf, D),$ $member(C, U1), triple(E, rdfs : subClassOf, C),$ $replace(U1, C, E, U2).$
(**Sub7**)	$triple(C, rdfs : subClassOf, inter(I2)) : -triple(C, rdfs : subClassOf, inter(I1)),$ $member(D, I1), triple(D, rdfs : subClassOf, E),$ $replace(I1, D, E, I2).$
(**Sub8**)	$triple(hasvalue(Q, O), rdfs : subClassOf, D) : -triple(Q, owl : subPropertyOf, P),$ $triple(hasvalue(P, O), rdfs : subClassOf, D).$
(**Sub9**)	$triple(exists(Q, C), rdfs : subClassOf, D) : -triple(Q, owl : subPropertyOf, P),$ $triple(exists(P, C), rdfs : subClassOf, D).$
(**Sub10**)	$triple(exists(P, E), rdfs : subClassOf, D) : -triple(E, rdfs : subClassOf, C),$ $triple(exists(P, C), rdfs : subClassOf, D).$
(**Sub11**)	$triple(C, rdfs : subClassOf, hasvalue(Q, O)) : -triple(P, owl : subPropertyOf, Q),$ $triple(C, rdfs : subClassOf, hasvalue(P, O)).$
(**Sub12**)	$triple(C, rdfs : subClassOf, forall(Q, D)) : -triple(Q, owl : subPropertyOf, P),$ $triple(C, rdfs : subClassOf, forall(P, D)).$
(**Sub13**)	$triple(C, rdfs : subClassOf, forall(P, E)) : -triple(D, rdfs : subClassOf, E),$ $triple(C, rdfs : subClassOf, forall(P, D)).$

where facts for predicates *class*, *property* and *individual* are defined for each atomic classes, (inverse and transitive) properties and individuals.

Table 4
Encoding of the Inference Calculus II

Rule Name	Prolog Rules
(**Type1**)	$triple(A, rdf : type, D) : -triple(C, rdfs : subClassOf, D),$ $triple(A, rdf : type, C).$
(**Type2**)	$triple(A, rdf : type, D) : -triple(inter(E), rdf : type, D),$ $triple\_cond(A, E).$
(**Type3**)	$triple(A, rdf : type, D) : -triple(exists(P, C), rdfs : subClassOf, D),$ $triple(B, rdf : type, C), triple(A, P, B).$
(**Type4**)	$triple(A, rdf : type, D) : -triple(hasvalue(P, O), rdfs : subClassOf, D),$ $triple(A, P, O), individual(O).$
(**Type5**)	$triple(B, rdf : type, D) : -triple(forall(P, C), rdfs : subClassOf, D),$ $triple(A, P, B), triple(A, rdf : type, C).$
(**Type6**)	$triple(A, rdf : type, D) : -triple(forall(inv(P), C), rdfs : subClassOf, D),$ $triple(A, P, B), triple(A, rdf : type, C).$
(**Type7**)	$triple(A, rdf : type, owl : thing) : -individual(A).$

Table 5
Encoding of the Inference Calculus III

Rule Name	Prolog Rules
(**Prop1**)	$triple(P, owl : equivalentProperty, P) : -property(P).$
(**Prop2**)	$triple(P, owl : equivalentProperty, R) : -triple(P, owl : equivalentProperty, Q),$ $triple(Q, owl : equivalentProperty, R).$
(**Prop3**)	$triple(P, owl : equivalentProperty, Q) : -triple(Q, owl : equivalentProperty, P).$
(**Prop4**)	$triple(P, owl : equivalentProperty, Q) : -triple(P, rdfs : subPropertyOf, Q),$ $triple(Q, rdfs : subPropertyOf, P).$
(**Prop5**)	$triple(P, rdfs : subPropertyOf, Q) : -triple(P, owl : equivalentProperty, Q).$
(**Prop6**)	$triple(P, rdfs : subPropertyOf, R) : -triple(P, rdfs : subPropertyOf, Q),$ $triple(Q, rdfs : subPropertyOf, R).$
(**Prop7**)	$triple(A, Q, B) : -triple(P, rdfs : subPropertyOf, Q), triple(A, P, B).$
(**Prop8**)	$triple(B, Q, A) : -triple(P, rdfs : subPropertyOf, inv(Q)), triple(A, P, B).$
(**Prop9**)	$triple(B, P, A) : -triple(inv(Q), rdfs : subPropertyOf, P), triple(A, Q, B).$
(**Prop10**)	$triple(A, P, C) : -triple(trans(P), rdfs : subPropertyOf, P),$ $triple(A, P, B), triple(B, P, C).$
(**Prop11**)	$triple(A, P, O) : -triple(C, rdfs : subClass, hasvalue(P, O)), triple(A, rdf : type, C).$

4.3 Using Prolog as Query Language

In this section, we will show how to use Prolog as query language for OWL. Basically, each query $\phi = \varphi_1, \ldots, \varphi_n$ in our query language can be encoded as a Prolog goal $? - triple(en(\varphi_1)), \ldots, triple(en(\varphi_n))$ in which each element of $\mathcal{V}_c$, $\mathcal{V}_p$ and $\mathcal{V}_i$ is encoded as a Prolog Variable. Now, we will show examples of queries against the ontology $\mathcal{O}_0$ and the corresponding answers. Let us remark

that in Prolog variables start with upper case letters.

Query 1: The first query we like to show is *"Retrieve the authors of manuscripts"*, which can be expressed in our query language as *Author_ of(a,b), Manuscript(b)*. It can be encoded as:

$$? - triple(A, author\_of, B), triple(B, rdf : type, manuscript).$$

Let us remark that our inference system is able to infer that a *Paper* and a *Book* is a *Manuscript* and therefore the above query retrieves all the manuscripts of the ontology $\mathcal{O}_0$. In addition, our inference system is able to infer that *author_ of* is a equivalent property to *writes*, and the inverse of *authored_ by*, and therefore all the cases are retrieved by the query language. In this case, the answers will be *"Abiteboul", "Buneman", "Suciu", "Simeon", "Buneman"* , together with names of manuscripts.

Query 2: The second query we would like to show is *"Retrieve the books of topic XML"* which can be expressed as $Book(book), topic(book, "XML")$. Now, it can be expressed as:

$$? - triple(Book, rdf : type, book), triple(Book, topic, "XML").$$

However, given that the ontology already includes the class "*XMLBook*" we can express the same query in a more concise way as $XMLBook(book)$ and therefore as:

$$? - triple(Book, rdf : type, xmlbook)$$

Query 3: The third query we would like to show is *"Retrieve the writers of reviewed manuscripts"*. It can be expressed as *Reviewed(manuscript), Writes(author, manuscript)*. In this case, the query can be expressed as:

$$? - triple(Manuscript, rdf : type, reviewed), triple(Author, writes, Manuscript).$$

Query 5: Let us see an example of query for retrieving meta-data from the ontology. For instance $range \sqsubseteq \forall Reviewed\_by.domain$ whose meaning is *"Find the domain and range of Reviewed_ by"* is encoded as:

$$? - triple(Range, rdfs : subClassOf, forall(reviewed\_by, Domain)).$$

In this case the answers will be *Manuscript, Book, Paper* for *Domain* and *Man, Woman, Person* for *Range*.

5 Conclusions and Future Work

In this paper we have proposed a query language for OWL based on Prolog. The query language is able to query about data and meta-data of a given ontology. As future work, we would like to extend our approach to richer fragments of DL and OWL.

In this line, the *W3C* has recently proposed a set of entailment rules for OWL, the so-called *OWL RL* [24]. Such set of rules subsumes our proposed entailment relationship and the pD^* approach. Therefore, our approach can be seen as a contribution to the OWL RL framework, providing an implementation by means of Prolog rules. We believe that our approach can be extended to OWL RL following the same technique here presented.

In addition, we would like to extend our work in the line of the SWRL [21] by incorporating Prolog rules to the reasoning with OWL, and therefore for enriching the defined query language.

References

[1] J. M. Almendros-Jiménez, A. Becerra-Terón, and Francisco J. Enciso-Baños. Magic sets for the XPath language. *Journal of Universal Computer Science*, 12(11):1651–1678, 2006.

[2] J. M. Almendros-Jiménez, A. Becerra-Terón, and Francisco J. Enciso-Baños. Querying XML documents in logic programming. *Theory and Practice of Logic Programming*, 8(3):323–361, 2008.

[3] J. M. Almendros-Jiménez, A. Becerra-Terón, and F. J. Enciso-Baños. Integrating XQuery and Logic Programming. In *Proceedings of the 17th International Conference on Aplications of Declarative Programming and Knowledge Management, INAP'07 and 21th Workshop on (Constraint) Logic Programming, WLP'07*, pages 117–135, Heidelberg, Germany, 2009. Springer LNAI, 5437.

[4] J. M. Almendros-Jiménez. An RDF Query Language based on Logic Programming. In *Proceedings of the 3rd Int'l Workshop on Automated Specification and Verification of Web Systems*. Electronic Notes on Theoretical Computer Science (200), 67–85, 2008.

[5] J. M. Almendros-Jiménez. An Encoding of XQuery in Prolog. In *Procs of the Sixth International XML Database Symposium XSym'09, at VLDB'09*, pages 145–155, Heidelberg, Germany, 2009. Springer, LNCS 5679.

[6] J. M. Almendros-Jiménez. Extending XQuery for Semantic Web Reasoning. In *Procs of the International Conference on Applications of Declarative Programming and Knowledge Management, INAP'09*, pages 109–124, 2009.

[7] J. M. Almendros-Jiménez. Ontology Querying and Reasoning with XQuery. In *Proceedings of the PLAN-X 2009: Programming Language Techniques for XML, An ACM SIGPLAN Workshop co-located with POPL 2009*. http://db.ucsd.edu/planx2009/papers.html, 2009.

[8] F. Baader, D. Calvanese, D.L. McGuinness, P. Patel-Schneider, and D. Nardi. *The description logic handbook: theory, implementation, and applications*. Cambridge Univ Press, 2003.

[9] Erol Bozsak, Marc Ehrig, Siegfried Handschuh, Andreas Hotho, Alexander Maedche, Boris Motik, et al. KAON - Towards a Large Scale Semantic Web. In *E-commerce and Web technologies: Third International Conference, EC-Web 2002*, pages 304–313. Springer, 2002.

[10] Dan Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, http://www.w3.org/TR/rdf-schema/, 2004.

[11] T. Berners-Lee, J. Hendler, O. Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[12] Jos de Bruijn, Rubén Lara, Axel Polleres, and Dieter Fensel. OWL DL vs. OWL flight: conceptual modeling and reasoning for the semantic Web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 623–632, New York, NY, USA, 2005. ACM.

[13] T. Eiter, G. Ianni, T. Krennwallner, and A. Polleres. Rules and ontologies for the semantic web. In *Reasoning Web*, pages 1–53. Springer, LNCS 5224, 2008.

[14] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proc. of the International Conference on World Wide Web*, pages 48–57, NY, USA, 2003. ACM Press.

[15] E. Grädel, P.G. Kolaitis, and M.Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

[16] W3C Working Group. OWL 2 Ontology Web Language. Technical report, http://www.w3.org/TR/owl2-overview/, 2009.

[17] A. Hogan, A. Harth, and A. Polleres. SAOR: Authoritative Reasoning for the Web. In *Proceedings of the 3rd Asian Semantic Web Conference on The Semantic Web*, pages 76–90. Springer-Verlag, 2008.

[18] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning in Description Logics by a Reduction to Disjunctive Datalog. *J. Autom. Reasoning*, 39(3):351–384, 2007.

[19] U. Hustadt, B. Motik, and U. Sattler. Deciding expressive description logics in the framework of resolution. *Information and Computation*, 206(5):579–601, 2008.

[20] V. Haarslev, R. Möller, and S. Wandelt. The revival of structural subsumption in tableau-based description logic reasoners. In *Proceedings of the 2008 International Workshop on Description Logics (DL2008), CEUR-WS*, pages 701–706, 2008.

[21] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, 21 May 2004. Available at http://www.w3.org/Submission/SWRL/.

[22] Graham Klyne and Jeremy J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, 2004.

[23] G. Lukacsy, P. Szeredi, and B. Kadar. Prolog based description logic reasoning. In *Proceedings of the 24th International Conference on Logic Programming (ICLP'08)*, pages 455–469, Heidelberg,Germany, 2008. Springer, LNCS 5366.

[24] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology: Reasoning in OWL 2 RL and RDF Graphs using Rules. Technical report, http://www.w3.org/TR/owl2-profiles/#Reasoning_in_OWL_2_RL_and_RDF_Graphs_using_Rules, 2009.

[25] S. Munoz, J. Pérez, and C. Gutierrez. Minimal deductive systems for RDF. In *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, page 53, Heidelberg, Germany, 2007. Springer, LNCS 4519.

[26] Boris Motik, Peter F. Patel-Schneider, and Bernardo Cuenca Grau. OWL 2 Web Ontology Language Direct Semantics. Technical report, http://www.w3.org/TR/owl2-direct-semantics/, 2009.

[27] Michael Schneider. OWL 2 Web Ontology Language RDF-Based Semantics. Technical report, http://www.w3.org/TR/owl2-rdf-based-semantics/, 2009.

[28] Evren Sirin, Bijan Parsia, Bernardo C. Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53, June 2007.

[29] Herman J. ter Horst. Extending the rdfs entailment lemma. In *International Semantic Web Conference*, pages 77–91. Springer, LNCS 3298, 2004.

[30] D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, pages 292–297. Springer, LNAI 4130, 2006.

[31] Raphael Volz. *Web Ontology Reasoning with Logic Databases.* PhD thesis, Universität Fridericiana zu Karlsruhe, 2004.

[32] J. Wielemaker, G. Schreiber, and B. J. Wielinga. Prolog-Based Infrastructure for RDF: Scalability and Performance. In *International Semantic Web Conference*, pages 644–658. Springer, LNCS 2870, 2003.