OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web

Jos de Bruijn
Digital Enterprise Research Institute (DERI)
University of Innsbruck, Austria

jos.debruijn@deri.org

Axel Polleres
Digital Enterprise Research Institute (DERI)
University of Innsbruck, Austria

axel.polleres@deri.org

Rubén Lara
Tecnología, Información y Finanzas
Madrid, Spain

rlara@afi.es

Dieter Fensel
Digital Enterprise Research Institute (DERI)
University of Innsbruck, Austria
National University of Ireland, Galway, Ireland

dieter.fensel@deri.org

ABSTRACT

The Semantic Web languages RDFS and OWL have been around for some time now. However, the presence of these languages has not brought the breakthrough of the Semantic Web the creators of the languages had hoped for. OWL has a number of problems in the area of interoperability and usability in the context of many practical application scenarios which impede the connection to the Software Engineering and Database communities. In this paper we present OWL Flight, which is loosely based on OWL, but the semantics is grounded in Logic Programming rather than Description Logics, and it borrows the constraint-based modeling style common in databases. This results in different types of modeling primitives and enforces a different style of ontology modeling. We analyze the modeling paradigms of OWL DL and OWL Flight, as well as reasoning tasks supported by both languages. We argue that different applications on the Semantic Web require different styles of modeling and thus both types of languages are required for the Semantic Web.

Categories and Subject Descriptors

I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation languages; H.3.4 [**World Wide Web**]

General Terms

Languages, Standardization

Keywords

Semantic Web, Ontologies, Description Logics, Logic Programming

1. INTRODUCTION

The vision of the Semantic Web [2] is to enable automatic interoperation between entities on the Web. Such interoperation can be achieved through annotation of the content on the Web with machine-processable data. When such annotations are linked to ontologies, machines can achieve a certain degree of understanding of the data. Ontologies [11] are formal and explicit specifications of certain domains and are shared between large groups of stakeholders. These properties make ontologies ideal for machine processing and enabling interoperation. In fact, ontologies form the backbone of the Semantic Web and are the key to enable automated interoperation and collaboration. An ontology typically consists of a number of classes, a number of relations (sometimes called properties) between these classes, a number of instances and a number of axioms. These elements are all expressed using some logical language.

In order to allow sharing and reuse of ontologies on the Semantic Web, a common ontology language is required. The W3C has developed two ontology languages for use on the Semantic Web. The first is RDFS [3], which was developed as a lightweight ontology language. The second language is OWL [7], which is a more expressive ontology language based on Description Logics [1].

OWL consists of three species, namely OWL Lite, OWL DL and OWL Full, which are intended to be layered according to increasing expressiveness. OWL Lite is a notational variant of the Description Logic $\mathcal{SHIF}(\mathbf{D})$; OWL DL is a notational variant of the Description logic $\mathcal{SHOIN}(\mathbf{D})$ [17]. It turns out that OWL DL adds very little in expressiveness to OWL Lite [19]. OWL Lite and OWL DL pose several restrictions on the use of RDF and redefine the semantics of the RDFS primitives; thus, OWL Lite and OWL DL are not properly layered on top of RDFS. The most expressive species of OWL, OWL Full, layers on top of both RDFS and OWL DL, and because these languages are so different, the semantics of OWL Full are not straightforward and are not a proper extension of the OWL DL semantics (see also Section 4.1). The lack of proper layering between RDFS and the less expressive species of OWL and the lack of proper layering between OWL DL and OWL Lite on the one side and OWL Full on the other, raises doubts about interoperability between ontologies written in these different languages.

For computing professionals from the areas of Software Engineering and Database Systems, there are some modeling pitfalls in OWL (see also [5]):

- Behavior of cardinality restrictions (the allowed number of values for a property): equality between individuals or existence of individuals outside the knowledge base can be inferred. The cardinality of properties is not *checked*, but *inferred*.

- Behavior of range restrictions: the type of property values can be inferred. The type of property values is not *checked*, but *inferred*.

Some of these potential pitfalls can be eliminated by introducing the unique name assumption and asserting complete knowledge about certain descriptions. The unique name assumption can be introduced in a Description Logic knowledge base by asserting inequality between each distinct pair of individual names. OWL DL offers the allDifferent operator for this purpose. For modeling complete knowledge, the Description Logic community offers the epistemic operator **K** [9]. When using the **K** operator as a prefix to a description, the description represents complete knowledge about the description, i.e., the knowledge base is assumed to contain all instances of this description.

In this paper we are not concerned with possible extensions of OWL or with the use of explicit assertions in order to enforce certain behavior of the modeling primitives. We are concerned with OWL DL as it is and with the modeling constructs provided by the language. In order to assert the unique name assumption in an OWL DL knowledge base, it is necessary to include all individuals in the knowledge base in the allDifferent statement. This is not feasible in practice. Clearly, OWL could be extended with a construct with the implicit meaning that all individuals are different and OWL could also be extended with a construct which asserts complete knowledge for a certain description. However, we examine in this paper the OWL language as it is and the properties of the OWL modeling constructs as they are being used today.

In order to investigate these modeling pitfalls of OWL and the usability of OWL for modeling and reasoning on the Semantic Web, we describe in this paper OWL Flight [6], an ontology language based on the Logic Programming subset of OWL [5] which is inspired by the intersection of Logic Programming and Description Logic [12] with certain extensions in the area of datatypes [26], database-style constraints and meta-modeling. Our two main motivations for creating OWL Flight were: (1) eliminate some of the pitfalls in conceptual modeling with OWL and (2) enable efficient query answering using common off-the-shelf reasoning engines which benefit from many years of research on optimizing query answering for deductive databases (e.g. [28]).

This paper is further structured as follows. We first discuss the formal and conceptual differences between restrictions and constraints in Section 2. Then, we briefly describe OWL DL and pitfalls in the use of this language for the Semantic Web in Sections 3 and 4, followed by the introduction of OWL Flight in Section 5. We analyze the conceptual modeling features of OWL DL and OWL Flight in the light of potential modeling tasks on the Semantic Web in Section 6. We contrast the reasoning tasks supported by the languages and the use cases for these reasoning tasks in Section 7. Finally, we provide some conclusions and mention future work in Section 8.

2. RESTRICTIONS AND CONSTRAINTS

In order to provide a better understanding of modeling and reasoning with different types of languages, we describe the difference between restrictions and constraints. The terms *restriction* and *constraint* both refer to some aspects of a property, such as the cardinality or the range of the property. We describe the differences first from a formal logical point of view and then from a conceptual modeling point of view.

For the purposes of our formal treatment of restrictions and constraints, we see an ontology as a logical theory where class definitions, property definitions, etc., are formulae in the theory. From a logical point of view, a restriction applied to a class definition can be seen as a first-order formula. Thus, a restriction *restricts* the number of models of a logical theory. Restrictions are thus an integral part of the logical theory. For an ontology Ω and a restriction r, let M^Ω be the set of models of Ω and $M^{\Omega \cup r}$ be the set of models of $\Omega \cup r$, then $M^\Omega \supseteq M^{\Omega \cup r}$. Let $cons(\Omega)$ be the set of consequences of Ω and $cons(\Omega \cup r)$ be the set of consequences of $\Omega \cup r$, then $cons(\Omega) \subseteq cons(\Omega \cup r)$. Thus, restrictions allow to *infer* additional information, because they increase the number of consequences.

Constraints[1] specify conditions which may not be violated by an interpretation of the logical theory. Constraints are not part of the logical theory and thus they do not increase the number of consequences of the theory. For ontology Ω and constraint c, if $\Omega \not\models c$ we say that the constraint is violated. However, since we don't see c as part of the theory, it does not affect the set of consequences. In summary, constraints do not allow to infer additional information, instead, they can be used to *check* the knowledge base with respect to certain conditions.

From a modeling point of view, restrictions and constraints are both used to specify certain aspects of a property, namely cardinality and range.

The *cardinality* of a property is the number of values a particular property may have for a particular individual. It is possible to specify a minimal bound and a maximal bound on the cardinality which we will refer to as minimal and maximal cardinality, respectively. Both minimal and maximal cardinality can be an arbitrary positive integer. Say we have a property P and an individual a. In case P has a minimal cardinality restriction of n, and the number of values of P for a in the knowledge base is smaller than n, a number of unknown individuals is inferred to exist outside of the knowledge base. In case P has a maximal cardinality restriction of n, and there are more than n values of P for a in the knowledge base, equality between property values is inferred in order to make the knowledge fit the restriction.

As for a minimal (or maximal, respectively) cardinality *constraint* of n on P, whenever the number of different values of P for a derivable from the knowledge base is smaller (or greater, respectively) than n, the constraint is violated and thus the knowledge base is erroneous.

The *range* of a property is the type a property value may have. Say we have a property P with a range restriction C and a tuple $\langle a, b \rangle \in P$. In case it is not known that b is of type C, it is inferred that b is of type C. In contrast, if P would have a range constraint C, then from the fact that it

---

[1] Our notion of constraints is similar to the notion of integrity constraints in logic programming and databases.

is not known that b is of type C, the constraint is violated and the knowledge base is inconsistent.

3. OWL DL

In this paper we are mainly concerned with the most well-known and most investigated species of OWL, namely OWL DL, which can be seen as an alternate notation for the Description Logic language $\mathcal{SHOIN}(\mathbf{D})$ [17].

OWL DL has different syntaxes, the most prominent being the RDF/XML syntax, which is actually used in the language reference. However, the normative syntax for OWL DL is the abstract syntax, described in [27], which we will use for the examples in the remainder of this paper for reasons of legibility. In the remainder of this section we will explain OWL DL using Description Logic syntax. See Tables 1 and 2 for a mapping between the OWL DL abstract syntax and the syntax of the Description Logic $\mathcal{SHOIN}(\mathbf{D})$.

OWL Abstract Syntax	DL syntax
Class axioms	
Class(A partial C_1 ... C_n)	$A \sqsubseteq C_i$
Class(A complete C_1 ... C_n)	$A \equiv C_1 \sqcap \ldots \sqcap C_n$
EnumeratedClass(A o_1 ... o_n)	$A \equiv \{o_1, \ldots o_n\}$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$
EquivalentClasses(C_1 ... C_n)	$C_1 \equiv \ldots \equiv C_n$
DisjointClasses(C_1 ... C_n)	$C_i \sqcap C_j \sqsubseteq \bot$
Property axioms	
ObjectProperty(R	
super(R_1)...super(R_n)	$R \sqsubseteq R_i$
domain(C_1) ... domain(C_n)	$\top \sqsubseteq \forall R^-.C_i$
range(C_1) ... range(C_n)	$\top \sqsubseteq \forall R.C_i$
[inverseOf(R_0)]	$R \equiv R_0^-$
[Symmetric]	$R \equiv R^-$
[Functional]	$\top \sqsubseteq\, \leqslant 1R$
[InverseFunctional]	$\top \sqsubseteq\, \leqslant 1R^-$
[Transitive])	Trans(R)
Datatype(T)	
DatatypeProperty(U	
super(U_1)...super(U_n)	$U \sqsubseteq U_i$
domain(C_1) ... domain(C_n)	$\top \sqsubseteq \forall U^-.C_i$
range(T_1) ... range(T_n)	$\top \sqsubseteq \forall U.T_i$
[Functional])	$\top \sqsubseteq\, \leqslant 1U$
SubPropertyOf(Q_1 Q_2)	$Q_1 \sqsubseteq Q_2$
EquivalentProperties(Q_1 ... Q_n)	$Q_1 \equiv \ldots \equiv Q_n$
Individual assertions	
Individual(o	
type(C_1) ... type(C_n)	$o \in C_i$
value(R_1 o_1) ... value(R_m o_m)	$\langle o, o_i \rangle \in Q_i$
value(U_1 t_1) ... value(U_n t_n))	$\langle o, t_i \rangle \in U_i$
SameIndividual(o_1 ... o_n)	$o_1 = \ldots = o_n$
DifferentIndividuals(o_1 ... o_n)	$o_i \neq o_j, i \neq j$

Table 1: Axioms in OWL DL and $\mathcal{SHOIN}(\mathbf{D})$

A Description Logic knowledge base consists of two parts, namely the TBox and the ABox. The TBox consists of a number of class and property axioms; the ABox consists of a number of individual assertions (see Table 1). Here, C refers to a description, T refers to a concrete datatype; D refers to either a description or a datatype. R refers to an object property name, U refers to datatype property; Q refers to an object or datatype property where several appearances of Q_i, Q_j in one statement always refer to either both object or both datatype properties; o and t refer to

object and concrete values, respectively. A class axiom in the TBox consists of two class descriptions, separated with the GCI (General Class Inclusion, or subsumption; $\sqsubseteq$) symbol or the equivalence symbol ($\equiv$), which is equivalent to GCI in both direction (i.e. $\sqsubseteq$ and $\sqsupseteq$). Similarly, a property axiom consists of two property names, separated with the subsumption ($\sqsubseteq$) or the equivalence ($\equiv$) symbol. A description in the TBox is either a named class (A), an enumeration ($\{o_1, \ldots o_n\}$), a property restriction ($\exists R.D$, $\forall R.D$, $\exists R.o$, $\geqslant nR$, $\leqslant nR$, analogously for datatype property restrictions), or an intersection ($C \sqcap D$), union ($C \sqcup D$) or complement ($\neg C$) of such descriptions (Table 2). Individual assertions in the ABox are either class membership ($o \in C_i$), property value ($\langle o_1, o_2 \rangle \in R_i$, $\langle o_1, t_1 \rangle \in U_i$), or individual (in)equality ($o_1 = o_2$, $o_1 \neq o_2$) assertions (Table 1).

OWL Abstract Syntax	DL syntax
A (URI Reference)	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf(C_1 ... C_n)	$C_1 \sqcap \ldots \sqcap C_n$
unionOf(C_1 ... C_n)	$C_1 \sqcup \ldots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf(o_1 ... o_n)	$\{o_1, \ldots o_n\}$
restriction(R someValuesFrom(C))	$\exists R.D$
restriction(R allValuesFrom(C))	$\forall R.D$
restriction(R value(o))	$\exists R.o$
restriction(R minCardinality(n))	$\geqslant nR$
restriction(R maxCardinality(n))	$\leqslant nR$
restriction(U someValuesFrom(T))	$\exists U.T$
restriction(U allValuesFrom(T))	$\forall U.T$
restriction(U value(t))	$\exists U.t$
restriction(U minCardinality(n))	$\geqslant nU$
restriction(U maxCardinality(n))	$\leqslant nU$

Table 2: Descriptions in OWL DL and $\mathcal{SHOIN}$

Description Logics have a set-based model-theoretic semantics. In an interpretation $\mathcal{I}$, a description C (Table 2) is mapped to a subset of the domain $\Delta^{\mathcal{I}}$ and an individual o is mapped to an object of $\Delta^{\mathcal{I}}$ using the mapping function $\cdot^{\mathcal{I}}$. Similarly, a datatype T is mapped to a subset of the concrete domain $\Delta_D^{\mathcal{I}}$ and a literal is mapped to a value in $\Delta_D^{\mathcal{I}}$. An abstract role R is mapped to a binary relation over the abstract domain domain: $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Similarly, a concrete role R is mapped to a binary relation between the abstract and the concrete domain: $\Delta^{\mathcal{I}} \times \Delta_D^{\mathcal{I}}$. Equivalence of descriptions is interpreted as set equivalence ($C \equiv D$ is interpreted as $C^{\mathcal{I}} = D^{\mathcal{I}}$), subsumption is interpreted as set inclusion ($C \sqsubseteq D$ is interpreted as $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$), and so on. We refer the interested reader to [1, Chapter 2] for a more exhaustive treatment of Description Logic semantics.

There exist several implementations for reasoning with Description Logics (e.g. FaCT++ [30], RACER [13]) which implement different reasoning tasks in Description Logic languages. Two important reasoning tasks in Description Logics are subsumption checking and checking class membership [1]. Subsumption checking amounts to checking whether one class is a subclass of another concept, i.e., checking whether one concept is more specific than another concept. The class membership inference is used to check whether an individual is a member of a specific class.

4. PITFALLS OF OWL

In this section we describe a number of (potential) pitfalls of OWL with respect to interoperability on the Semantic Web and scalability of reasoning with the language. Furthermore, we discuss the suitability of its modeling constructs and modeling style for certain domains and extensibility of the language in the direction of rules.

4.1 Interoperability

Problems of interoperability might occur between RDFS and OWL and between the species of OWL because of problems in the layering of the languages.

The OWL language is layered on top of RDFS. However, only the most expressive species of OWL, namely OWL Full, is completely syntactically and semantically layered on top of RDFS. The less expressive species of OWL, namely OWL Lite and OWL DL, pose syntactical restrictions on the use of RDF and redefine the semantics of the RDFS modeling primitives.

The species of OWL are layered according to increasing expressiveness, where OWL Lite is the least expressive and OWL Full the most expressive. On the one hand, OWL Lite poses many syntactical restrictions on the constructs which can be used in ontology modeling. On the other hand, the only feature really added by OWL DL compared with OWL Lite is the use of *nominals* (individuals in class descriptions) [19]; all other features of OWL DL can be written down using OWL Lite through complicated syntactical constructions. The third, and most expressive, species of OWL, namely OWL Full, is unfortunately not properly semantically layered on top of OWL DL; entailment under OWL DL semantics is not equivalent to entailment under OWL Full semantics for the same ontology: OWL Full allows additional inferences. This discrepancy is caused by the incompatibility between the model-theoretic semantics of OWL DL and the axiomatic semantics of and syntactical freedom of RDFS. This raises doubts about the level of interoperability between the different species of OWL.

Both the layering of OWL on top of RDFS and the layering of the OWL species (especially the layering of OWL Full on top of OWL DL) is, in our opinion, inappropriate. The two less expressive species of OWL, OWL Lite and OWL DL, are only layered on top of a restricted subset of RDFS, whereas the most expressive species of OWL, OWL Full, is completely syntactically and semantically layering on top of RDFS, but not on top of OWL Lite and OWL DL. This improper layering might hamper interoperability between agents using RDFS or OWL Full on the one side and agents using OWL Lite or OWL DL on the other.

4.2 Scalability

There are doubts with respect to the scalability of certain reasoning tasks in OWL, mostly query answering. Description Logic reasoning and optimization has so far mostly focused on the optimization of the subsumption inference; few optimizations exist for query answering. Optimizing query answering for Description Logics is currently still very much a research issue [12, 22, 14, 16].

The satisfiability problem in $\mathcal{SHOIN}$, the Description Logic underlying OWL DL, has NExpTime (non-deterministic exponential time) worst-case complexity. Most current Description Logic implementations use complex Tableaux satisfiability checks for query answering [21]. The major problem with using Tableaux for query answering for knowledge bases with more than a few individuals is that a Tableaux check is required for each individual in the knowledge base to check whether it is in the answer to the query. Although there exist several optimizations [14, 16], the fundamental problems are not solved. In our opinion it is a mistake to require such complex reasoning for even the least expressive of the OWL species, since efficient ABox reasoning will most likely play a major role on the Semantic Web. There are currently over four billion web pages indexed by Google, thus, in order for the Semantic Web to work outside of the research lab, it must be possible to reason with large collections of instances.

4.3 Conceptual Modeling

Some of the modeling constructs of OWL DL have a semantics which might seem odd to people not familiar with Description Logics. These constructs concern the treatment of abstract vs. concrete values, cardinality restrictions, and value restrictions. Furthermore, we identify limitations in the support for datatypes in OWL.

Difference in the treatment of abstract and concrete values. The interpretation of literals (concrete values) in OWL is fixed and thus unknown equality between literals can not be derived. Consequently, cardinality and range restrictions exhibit different behavior depending on whether they are concerned with object or datatype properties. Restrictions over object properties exhibit the usual behavior of restriction as we have described in Section 2. Restrictions over datatype properties, however, exhibit the behavior of constraints as described in Section 2. It has been argued that the difference in treatment of individuals and literals makes sense, because the domain of a data type is known. We argue that from the point of view of the user of the language, this conceptual distinction is not intuitive. Whereas restrictions involving the abstract domain are used to *infer* new knowledge, restrictions involving the concrete domain are used to *check* whether the knowledge satisfies certain constraints.

Deriving Equality through Cardinality Restrictions. In OWL, it is possible to specify a maximal cardinality restriction for a property. When there are more instances of this property with the same domain value than the maximal cardinality restriction prescribes, equality between individuals is inferred. We illustrate this with an example.

EXAMPLE 1. *Assume the following OWL DL knowledge base:*
```
ObjectProperty(hasPassenger domain(FlightSeat)
  range(Passenger))
Class(FlightSeat partial
  restriction(hasPassenger maxCardinality(1)))
Individual(seat1 type(FlightSeat)
  value(hasPassenger mary)
  value(hasPassenger john))
```

FlightSeat represents the seats in a particular flight. The property hasPassenger associates a seat in a flight with a passenger. A seat may only have one passenger, which is guaranteed by the restriction maxCardinality(1). The individual seat1 an instance of FlightSeat; seat1 has two values for the property hasPassenger, namely mary and john.

From the OWL knowledge base in Example 1 the reasoner will draw the conclusion that `mary` and `john` both refer to the same passenger. The possibility that there was a mistake in the system (either at the ontology or at the instance level) is not taken into account. This could lead to the booking of several passengers on a single seat, unless the unique name assumption is enforced.

Deriving Class Membership through Value Restrictions. In OWL it is possible to restrict the range of a property P to a class description C either through a range restriction in the property definition or through a local universal range restriction in a class definition. From this restriction it is inferred that every value of this property is a member of class C.

EXAMPLE 2. *We take the OWL knowledge base from Example 1 and add the following assertions:*

```
Individual(seat3 type(FlightSeat))
Individual(seat2 type(FlightSeat)
value(hasPassenger seat3))
```

The above assertions introduce two new instances of the class `FlightSeat` *by the names of* `seat2` *and* `seat3` *plus a value for the property* `hasPassenger` *at* `seat3`, *namely* `seat3`.

From Example 2 we can infer that `seat3` is a `Passenger`. Clearly, there is some mistake in the individual assertions, because a seat cannot occupy another seat; only a passenger can. However, this mistake is not detected; instead the modeling mistake allows for additional (incorrect) inferences. Although it is possible in OWL DL to express disjointness of classes, in which case an inconsistency would be derived, we argue that in many application domains it is natural to assume disjointness of classes beforehand and only deviate from this assumption when classes are *known* not to be disjoint.

Limited Support for Datatypes. OWL allows for a limited treatment of datatypes. Only unary datatypes are supported by OWL. The three major limitations of datatype support in OWL are [26]: (1) lack of negated datatypes, which is required for most Description Logic reasoners, (2) lack of support for datatype predicates; it is only possible to refer to a single value in a datatype domain and not, e.g., to express the greater-than ($\geq$) relation for the `xsd:integer` domain, and (3) lack of support for user-defined datatypes.

OWL-E [26] is an extension of OWL with so-called datatype groups. Datatype groups overcome the aforementioned limitations of datatype support in OWL and bridge the gap between datatypes in OWL and concrete domains as they have been investigated in the Description Logic community (see e.g. [20]).

4.4 Extensibility

It has been suggested that besides an ontology language, the Semantic Web needs a rule language. Such a rule language would provide additional expressiveness on top of the ontology language. It was identified as a requirement that the rule language is an extension of the ontology language.

There have been several proposals for rule extensions of Description Logic languages (e.g., CARIN [24] and SWRL [18]).

SWRL (Semantic Web Rule Language) has been proposed as a rule language for the Semantic Web, layered on top of OWL DL. Satisfiability of an OWL DL knowledge base augmented with SWRL rules is undecidable, as was pointed out by the authors of the proposal [18]. Straightforward extension of Description Logics with rules leads to undecidability [24] and does not allow the use of existing rule systems to reason with the language. Instead, complex new calculi need to be developed, or first-order theorem proving is required.

5. OWL FLIGHT

As opposed to OWL DL, in this section we will define a novel variant of OWL, called OWL Flight, which addresses some of the above-mentioned problems. On the one hand, OWL Flight restricts the OWL syntax such that it falls in the Datalog fragment and thus query answering can be done using a Logic Programming implementation (cf. [12]). On the other hand, we take this restricted subset of OWL DL as a basis which we further extend in order to overcome some of the limitations of OWL DL. These extensions include integrity constraints, a more elaborate treatment of datatypes than OWL based on OWL-E [26], and meta-modeling features where we make use of the meta-modeling features of the Datalog compatible variant of F-Logic [23].

The formalism underlying OWL Flight is Datalog$^{IC, \neq, not}$, which is Datalog extended with integrity constraints, inequality and default negation. OWL Flight is an extension of the so-called DLP fragment of OWL which marks the intersection of Description Logics and Datalog [12, 5]. Compared to DLP, we further add limited support for nominals, as well as a meta-modeling facility (e.g., treating classes as instances) and constraints.

Features of OWL DL not included in OWL Flight are: enumerated classes, individual (in)equality assertions, complements, property restrictions in complete class definitions. Furthermore, we restrict the use of property restrictions, nominals, and union on the left- and right-hand-side of the GCI.

OWL Flight adds the following distinct features to this restricted subset of OWL DL: We adopt the *unique name assumption*[2] and adding *cardinality constraints*, i.e., checking cardinalities rather than the non-intuitive inference of equality. *Property value constraints* eliminate the non-intuitive inferring of class membership. Constraints adopt the closed world assumption, allowing to check the data in the knowledge base, i.e. a single ontology (along with imported ontologies); as mentioned above, this feature is useful in many contexts and missing in OWL DL (cf. [15, 9]). *Meta-modeling* returns modeling support lost in the transition from RDFS to OWL DL. *More elaborate treatment of datatypes*, following OWL-E [26], which overcomes the limitations of the treatment of datatypes in OWL.

5.1 OWL Flight Abstract Syntax and Mapping to F-Logic

Syntactically, we base OWL Flight on a subset of the abstract syntax of OWL DL, but add distinct constructs

---

[2]Notice that also the prominent Description Logic reasoner RACER assumes unique names, not implementing equality reasoning, for efficiency reasons. As pointed out above, we do not necessarily see the lack of equality reasoning as a drawback but even as an advantage, due to unintuitive effects it might have with respect to reasoning in OWL DL.

for constraints and eliminate the separation of the vocabulary. Table 3 shows a feature comparison of OWL DL and OWL Flight based on an extended version of the OWL abstract syntax. The meaning of the letters in the table corresponds with the description given in Section 3; furthermore, E stands for a datatype predicate and n stands for the arity of the datatype predicate. The terms lhs (and rhs, resp.) in the table express that certain descriptions in OWL Flight are restricted to be used in the left-hand side (or right-hand side, resp.). Descriptions allowed on the right-hand side are allowed in `partial` class definitions and in the second argument of `SubClassOf`. Descriptions allowed on the left-hand side are allowed in the first argument of `SubClassOf`. Descriptions allowed on both sides are also allowed in `complete` class definitions. On the one hand, unrestricted usage of these features would lead us outside the expressivity of Datalog$^{IC,\neq,not}$. On the other hand, some of the unintuitive features of OWL DL like inferring equality originate precisely from this unrestricted usage. In the lower part of Table 3 we find the features newly added in OWL Flight, which are constraints in partial class definitions and Datatype expressions. As described in Section 2, constraints check all inferred instances of the respective class on the lhs for the condition on the rhs. Datatype expressions follow the idea of [26], which allows to define n-ary datatype predicates which constrain the values of the tuples given by datatype properties $U_1, \ldots, U_n$ of an object. So, with this extension, one can express relations between datatype properties, for instance that the number of bookings for a hotel room has to be smaller than its capacity, etc.

Note that constraints are only allowed for object properties, because for datatype properties the `restriction` keyword in OWL DL already has a constraining semantics as discussed in Section 4. For the sake of brevity, we refer to [6] for a complete description of the abstract syntax of OWL Flight.

We define the semantics of OWL Flight through a mapping to the Datalog subset of F-Logic [23]. By doing so, we gain the following benefits: Staying within the Datalog world, we can directly use implemented engines tailored for efficient query answering. F-Logic syntax with its origins in Frame based modeling (as opposed to pure Datalog) directly allows for meta-modeling features and provides constructs for class membership, subclassing and attributes in the language, rather than relying on predicates to axiomatize the behavior of these constructs. The complete mapping is defined in [6] and we restrict ourselves to the rough idea in this paper.

Let x, y, z be variables, : stands for class membership, :: stands for the subclass relationship and a molecule of the form $A[B{\twoheadrightarrow}C]$ stands for "object A has an attribute B with value C". Such molecules can be combined, for instance $r1 : Room[hasBooked{\twoheadrightarrow}2]$, stating that $r1$ is an instance of the concept Room, having the value 2 for attribute $hasBooked$. Over such molecules we define clauses of the form

$$m \leftarrow m_1, \ldots, m_n$$

with the usual meaning from logic programming, where the body molecules of such rule may be preceded by the negation as failure symbol `not`. Furthermore we allow modules of the form $x \neq y$, representing inequality, in the body of such clauses. Variables in F-Logic might be used for any kind

Abstract Syntax	DL	Fl.
Axioms and Individual Assertions		
`Class(`A `partial` C_1 ... C_n`)`	+	+
`Class(`A `complete` C_1 ... C_n`)`	+	+
`EnumeratedClass(`A o_1 ... o_n`)`	+	−
`SubClassOf(`C_1 C_2`)`	+	+
`EquivalentClasses(`C_1 ... C_n`)`	+	+
`DisjointClasses(`C_1 ... C_n`)`	+	+
`ObjectProperty(`R `...)`	+	+
`Datatype(`T`)`	+	+
`DatatypeProperty(`U `...)`	+	+
`SubPropertyOf(`Q_1 Q_2`)`	+	+
`EquivalentProperties(`Q_1 ... Q_n`)`	+	+
`Individual(`o `type(`C_1`)` ... `type(`C_n`)`	+	+
`value(`R_1 o_1`)` ... `value(`R_n o_n`))`	+	+
`SameIndividual(`o_1 ... o_n`)`	+	−
`DifferentIndividuals(`o_1 ... o_n`)`	+	−*
Descriptions		
A (URI Reference)	+	+
`owl:Thing`	+	+
`owl:Nothing`	+	+
`intersectionOf(`C_1 ... C_n`)`	+	+
`unionOf(`C_1 ... C_n`)`	+	lhs
`complementOf(`C`)`	+	−
`oneOf(`o_1 ... o_n`)`	+	lhs
`restriction(`Q `someValuesFrom(`D`) )`	+	lhs
`restriction(`Q `allValuesFrom(`D`))`	+	rhs
`restriction(`Q `value(`o`)`	+	+
`restriction(`Q `minCardinality(`n`))`	+	−
`restriction(`Q `maxCardinality(`n`))`	+	rhs
`constraint(`R `someValuesFrom(`C`))`	−	rhs
`constraint(`R `allValuesFrom(`C`))`	−	rhs
`constraint(`R `value(`o`))`	−	rhs
`constraint(`R `minCardinality(`n`))`	−	rhs
`constraint(`R `maxCardinality(`n`))`	−	rhs
Datatype Expressions and n-ary Datatype constraints		
`DatatypeExpression (`E/n		
`and/or/domain(` E_1/n ...E_m/n `) )`	−	+
`restriction(`U_1_U_n `someValuesFrom(`E/n`))`	−	lhs
`restriction(`U_1_U_n `allValuesFrom(`E/n`))`	−	rhs

* would be superfluous because OWL Flight adopts UNA

Table 3: Features of OWL DL vs. OWL Flight

of objects, individuals, concepts or even attributes, allowing meta-modeling with sets of such clauses (i.e., F-Logic Programs). There exist efficient engines, such as Ontobroker [8] and FLORA-2 [32], for evaluating these kinds of programs. Both implementations use a translation to plain Datalog in order to evaluate the program using a plain Datalog engine.

We will illustrate the mapping from OWL Flight to F-Logic with some small examples. Simple subclass axioms or partial class definitions for named classes, i.e., statements of the form `Class(`A `partial` C_1 ... C_n`)` are translated to sets of statements of the form:

$$\bigwedge A :: C_i$$

Nested property restrictions in C_i can be handled by chaining variables, e.g.
`Class(`A `partial restriction(R allvaluesFrom`
  `(restriction S allValuesFrom B)))`
is translated to:

$$z : B \leftarrow x : A, x[R{\twoheadrightarrow}y], y[S{\twoheadrightarrow}z]$$

We translate constraints to integrity constraints, i.e., clauses with an empty head, and involve negation as failure for checking integrity on the knowledge base. So, e.g.

Class(A partial constraint(R allValuesFrom(C)))
will result in

$$\leftarrow x : A[R {\twoheadrightarrow} y], \text{not } y : C$$

The complete translation [6] involves some particularities, but it allows us to translate all of OWL Flight to function-free F-Logic programs with integrity constraints and default negation, i.e. the Datalog$^{IC, \neq, not}$ fragment of F-Logic. The Datatype-Expression facilities of OWL-E fit nicely in this translation.

Summarizing, OWL Flight on the one hand uses a maximal subset of OWL DL which is compatible with the logic programming world and on the other hand defines slight extensions addressing some of the limitations identified in Section 4. Although this new OWL "dialect" can not to be viewed as a fully-fledged ontology language to solve all the above-mentioned problems, we conceive it as a solid starting point for a unified framework of OWL based ontology languages combining the benefits of the Description Logics and Logic Programming paradigms. Furthermore, it serves as a good starting point to investigate the limitations of OWL and ways to overcome its limitations.

6. CONCEPTUAL MODELING ON THE SEMANTIC WEB

In this section we compare the modeling constructs of OWL Flight and OWL DL in the context of modeling tasks on the Semantic Web.

6.1 Cardinality Restrictions and Constraints

OWL DL has the notion of *cardinality* restrictions, which can be used in class definitions. Cardinality restrictions allow for inferring equality and/or the existence of individuals not in the knowledge (see also Section 2). OWL Flight has an explicit notion of cardinality constraints, which are used to *check* the number of values for a certain property, rather than to derive equality or assume existence of individuals beyond the knowledge base.

Besides the possible non-intuitiveness of deriving equality (see also Section 4.3), we see the following pitfall for a language which allows to infer equality: The possibility to derive equality might be misleading, because not all equality on the Semantic Web can be resolved in the logical language.

Equality reasoning in OWL DL is not powerful enough to resolve all equalities between identifiers on the Semantic Web. Only if the world would be perfectly and completely modeled in an ontology and only if all individuals on the Semantic Web are related to this ontology could all equalities on the Semantic Web be resolved. We argue that very few equalities can actually be resolved with reasoning and that many derived equalities are actually faulty. Thus, it makes more sense in our opinion to either resolve equalities beyond the logical language or to make strong assumptions on the available knowledge, i.e., assume that each identifier in the knowledge base uniquely identifies an individual (the *unique name assumption*).

6.2 Value Restrictions vs. Constraints

OWL allows for two kinds of value restrictions, namely, existential and universal. However, an existential value restriction merely corresponds to a qualified minimal cardinality of 1. Therefore, and because we have already discussed cardinality restrictions in the previous section, we only treat universal value restrictions here.

When used in a class definitions, a value restriction over an object property can be used to derive additional information about property values with members of this class as its domain (see also Section 4.3).

OWL Flight allows for the specification of value *constraints*. A constraint is used to *check* whether the knowledge in the knowledge base conforms to the constraint. Take for example the following definition:

Class(Parent partial
    constraint(hasChild allValuesFrom Person)

This constraint does not allow any child which is not known to be a person.

We argue that the constraining approach is more intuitive to many computing professionals with Software Engineering and Database Systems backgrounds [5]. However, constraints do not allow additional inferences. Therefore, there is a tradeoff between the ability to detect modeling mistakes and the inferencing power of the language. OWL DL focuses more on the inferencing power of the language, whereas OWL Flight focuses more on catching modeling mistakes.

We have pointed out in Section 4.3 that in OWL DL there is an asymmetry between the treatment of datatype and abstract individuals. From a modeling point of view, this asymmetry is most apparent in the way property restrictions and range restrictions of datatype properties are handled, as we have argued in Section 4.3. Consider the following OWL DL knowledge base:

Class(A partial restriction(Q allValuesFrom D))
Individual(a value(Q b))

Say, Q is an object property and D is an abstract class description. By the universal value restriction, we can infer that b is an instance of D.

Now suppose Q is a datatype property and D is a datatype. From this knowledge base, we cannot conclude that b is a data value of type D, and thus the knowledge base is inconsistent.

This asymmetry between the treatment of object and datatype properties does not occur in OWL Flight, because it allows to model constraints for object properties which are treated in the same way as constraints for datatype properties.

6.3 Meta-modeling

Meta-modeling is based on the principle that the same object can be seen as a class, an individual or a property, depending on the point of view. An example taken from [29] is the object 'Boeing747' which is an instance of the class 'AircraftType' but is itself also a class whose instances are the individual aircraft. Another example [25] is the description of the topics of individual books. Each topic is a class and each book is an individual. This obviates the need to use classes as property values. Because of the heterogeneity to be expected on the Semantic Web, a conceptual

modeling language for the Semantic Web should support meta-modeling.

RDFS and OWL Full allow full meta-modeling. Each identifier can denote a class, instance and/or property. Unfortunately, RDFS is not expressive enough for many applications and OWL Full is undecidable in general. The decidable species of OWL, namely OWL Lite and OWL DL, unfortunately do not allow meta-modeling. Instead, they require a strict separation between identifiers of classes, individuals and properties.

In the case of a strict separation between classes and instances, it is sometimes hard to find an agreement as to whether to model an object as a class or an instance. Since an ontology is shared by its very nature, it is important to find this agreement. In case inter-operation is required between different ontologies, this requirement becomes even more apparent, because the modelers of the different ontologies might have different notions of what is an instance and what is a class.

OWL Flight allows meta-modeling while retaining decidability, following the meta-modeling support of F-Logic [23], which allows a limited form of meta-modeling by interpreting an identifier differently depending on the context in which it occurs. In this way, F-Logic stays inside the expressiveness of first-order logic. OWL Flight allows meta-modeling in the same way as F-Logic and by doing this, it stays inside the (decidable) Datalog fragment.

6.4 Complete Class Definitions

OWL allows two types of class definitions: partial class definitions and complete class definitions. A partial class definition corresponds with a necessary definition, thus, it specifies all conditions (such as superclasses and property restrictions) which are necessarily fulfilled (and thus *inferred*) for all members of the class. A complete class definition specifies necessary and sufficient conditions, which means that not only are these conditions inferred from class membership, but class membership is also inferred if all conditions are fulfilled, i.e., if an individual is a member of all superclasses and all property restrictions are fulfilled.

Complete class definitions are allowed only to a limited extent in OWL Flight: only named classes and individual value restrictions are allowed in the definition (see also Table 3). In contrast, OWL DL allows all descriptions in both partial and complete class definitions.

To harvest the full power of Description Logic reasoning, complete class definitions should be used, because they allow for more powerful inference than partial definitions. However, it is hard to model complete definitions correctly, especially because it is easy to miss certain aspects of a class when creating the definition, in which case the definition is incorrectly labeled as 'complete'. Thus, there is again a tradeoff between the ability to detect modeling mistakes and the inferencing power of the language.

Correct and complete modeling cannot be guaranteed in general in such an open and distributed environment as the Web. However, in closed and controlled sections of the Semantic Web where it is possible to guarantee a certain degree of correctness of modeling, complete class definitions can be very useful. Complete class definitions are related to subsumption reasoning, to be discussed in Section 7.1.

7. REASONING TASKS ON THE SEMANTIC WEB

In this section we describe the primary reasoning tasks on the Semantic Web and how there are supported by OWL DL and OWL Flight. The primary reasoning tasks we consider are subsumption and query answering.

7.1 Subsumption

Subsumption corresponds with checking whether a class description subsumes (is more general than) another class description. By doing this for all classes in the knowledge base, one can compute the subsumption hierarchy. By checking the place in the subsumption hierarchy of a given class description, this class description is classified with respect to the knowledge base and hidden relationships with other classes in the knowledge base become visible.

Subsumption can be reduced to checking (un)satisfiability and thus, for OWL DL, an efficient Description Logic algorithm [21] can be used to compute the subsumption for any two OWL DL descriptions. There exist several implementations for the subset of OWL DL which excludes nominals, such as RACER [13] and FaCT++ [30].

For OWL Flight, subsumption reasoning can be performed using a Description Logic reasoner for that subset of OWL Flight which falls inside OWL DL. There are also techniques for subsumption reasoning using Logic Programming engines, again using the same fragment of OWL Flight. These techniques [12, 31] effectively reduce subsumption reasoning to query answering in a deductive database. However, Description Logic reasoners are in general more efficient for subsumption reasoning (cf. [31]).

It is not entirely clear how constraints in OWL Flight interact with subsumption reasoning. Presumably, they should not be taken into account when doing subsumption reasoning, because constraints are not part of the logical theory (see also Section 2). This is a matter which requires further investigation.

Subsumption reasoning allows inferring relationships between terms which have not been explicitly stated. On the one hand, this kind of reasoning might not be easily accepted in some areas. We believe that businesses will want control over their business vocabularies and do not want (possibly incorrect) inferred relationships between terms.

On the other hand, some areas may greatly benefit from the additional knowledge derived in subsumption reasoning. For example, search results in Knowledge Management applications might be improved because more relevant terms are taken into account and also other applications where 100% precision is not required or where complete and correct modeling can be assumed and where mistakes introduced through inferences can be tolerated can benefit greatly from classification reasoning.

There is a tradeoff between the possibilities for automation on the one hand and the requirements on the completeness and correctness of the modeling on the other hand. If one wants to automate certain tasks on the Semantic Web, such as Web Service execution, then correct and complete modeling of goals and web services is required. If one cannot assume correct and complete modeling of the functionality of a web service, then it is impossible to automate execution.

	Conceptual Modeling	Reasoning tasks
OWL DL	• Restrictions allow inferring new knowledge • Modeling style of Knowledge Representation • Complete class descriptions	• Efficient subsumption for a significant part (SHIQ) • Query Answering under-developed
OWL Flight	• Constraints check existing knowledge • Modeling style of Databases and Software Engineering • Limited complete class descriptions	• Limited expressiveness for classification; not optimized • Query Answering well-developed; many well-known optimization techniques and implementations

Figure 1: OWL DL and OWL Flight: conceptual modeling and reasoning

7.2 Query Answering

There exist two different types of queries, namely, ground queries and open queries. A ground query consists of a knowledge base and a ground fact. The inference task is to check whether the ground fact is entailed by the knowledge base. An open query is a formula with free variables. The query answer consists of a number of substitutions for the variables with values from the knowledge base. An open query can be reduced to a number of ground queries, namely, a ground query for each of the facts in the knowledge base. This is clearly not efficient and thus optimizations have been (and are being) developed for answering such open queries.

As we can see from Section 7.1, the main power of Description Logics lies in subsumption reasoning. Query answering reasoning is also possible, but is in general less investigated and very few optimizations exist to do query answering efficiently in Description Logics, as we have argued in Section 4.2.

OWL Flight was developed with efficient query answering using deductive databases in mind. Thus, common off-the-shelf deductive database engines (e.g. OntoBroker [8], FLORA-2 [32]) can be used for reasoning with OWL Flight. These deductive databases incorporate optimizations which have been developed in the database research area.

Summarizing, we can say that the expressive power, the modeling constructs and the current state-of-the-art reasoners for Description Logics (and thus OWL DL) are tuned to powerful subsumption reasoning. The expressive power of OWL Flight takes into account current state-of-the-art deductive databases and thus allows to take advantage of efficient query answering. OWL DL has significantly more expressive power than OWL Flight, because of disjunction and existential quantification allowed in the language. Therefore, even if optimizations for query answering with Description Logics are developed, these implementations cannot get around the theoretical complexity of the reasoning task.

8. CONCLUSIONS

In this paper we have introduced OWL Flight, a variant of OWL which is based on Logic Programming rather than Description Logics. We took Description Logic Programs [12] (the intersection of Description Logics and Logic Programming) as a basis and extended it with cardinality and value constraints, meta-modeling and powerful datatype support (based on OWL-E [26]).

We have compared the modeling styles for OWL DL and OWL Flight, as well as the reasoning tasks supported by both languages. We summarize the modeling styles and reasoning tasks for both languages in Figure 1.

With respect to the modeling styles we can conclude that OWL DL has an inferring modeling style. Restrictions on property values and cardinalities are used by the reasoner to *infer* new knowledge from the existing knowledge, such as equality of individuals and class membership. OWL Flight has a constraining modeling style; it allows expressing cardinality and values constraints on properties. Such constraints are used by the reasoner to *check* whether the knowledge in the knowledge base corresponds with the conditions expressed in the constraints.

With respect to the supported reasoning tasks, we can conclude that OWL DL is most suited for the subsumption task and OWL Flight is most suited for the query answering task, because of the modeling styles of the languages and because of the optimized algorithms and implementations which have been developed for the respective modeling tasks. There exists a tradeoff between the inferencing power of a language and the ability to detect modeling mistakes.

We can conclude that different tasks on the Semantic Web need different styles of modeling and different types of reasoning. Therefore, we argue that it is not sufficient to restrict the Semantic Web to only one style of modeling for ontologies, which is currently done by OWL, in the form of Description Logics. Many applications on the Semantic Web would benefit from an ontology language which favors a constraining modeling style, such as OWL Flight. Furthermore, such a language might appeal more to software engineers and database designers, so that they more easily adopt the Semantic Web.

Interoperation between Description Logic-based and Logic Programming-based languages can be achieved either through their common subset [12], through an interface between the languages [10] or through a common superset. Such a common superset is not easy to define, because the default negation common in logic programming does not fit nicely in a first-order framework. Nonmonotonic extensions such as the **K** operator [9] could help bridge the gap.

We argue that the Semantic Web requires a unifying framework for the Logic Programming and the Description Logic paradigm. Such a unifying framework would consist of a core language, based on [12], which is the common subset of both paradigm. The framework should have extensions in both the Logic Programming and the Description Logic directions to allow applications on the Semantic Web to take advantage of existing algorithms and implementations. The languages would then be unified in a language which captures both paradigms. This unifying language could be a first-order language with nonmonotonic extensions in order to capture the nonmonotonicity of negation in Logic Programming. We are currently in the process of developing such a framework in the context of the WSML Working Group [4].

9. REFERENCES

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[3] D. Brickley and R. V. Guha. RDF vocabulary description language 1.0: RDF schema. Recommendation 10 February 2004, W3C, 2004.

[4] J. de Bruijn, H. Lausen, and D. Fensel. The WSML Family of Representation Languages. Deliverable D16.1v0.2, WSML, 2004. http://www.wsmo.org/TR/d16/d16.1/v0.2/.

[5] J. de Bruijn, A. Polleres, R. Lara, and D. Fensel. OWL⁻. Deliverable D20.1v0.2, WSML, 2004. http://www.wsmo.org/TR/d20/d20.1/v0.2/.

[6] J. de Bruijn, A. Polleres, R. Lara, and D. Fensel. OWL flight. Deliverable D20.3v0.1, WSML, 2004. http://www.wsmo.org/TR/d20/d20.3/v0.1/.

[7] M. Dean and G. Schreiber, eds. *OWL Web Ontology Language Reference*. 2004. W3C Recommendation 10 February 2004.

[8] S. Decker, M. Erdmann, D. Fensel, and R. Studer. *Ontobroker: Ontology based Access to Distributed and Semi-Structured Information*. Kluwer Academic, 1999.

[9] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. An epistemic operator for description logics. *Artificial Intelligence*, 100(1–2):225–274, April 1998.

[10] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the semantic web. In *Proc. of the Int. Conf. of Knowledge Representation and Reasoning (KR2004)*, 2004.

[11] D. Fensel. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce, 2nd edition*. Springer-Verlag, Berlin, 2003.

[12] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proc. Intl. Conf. on the World Wide Web (WWW2003)*, Budapest, Hungary, 2003.

[13] V. Haarslev and R. Möller. Racer system description. In *Proc. of the 12th Int. World Wide Web Conf. (WWW2003)*, pages 48–57. ACM, 2003.

[14] V. Haarslev and R. Möller. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Proc. of the Int. Conf. of Knowledge Representation and Reasoning (KR2004)*, pages 163–173, 2004.

[15] J. Heflin and H. Muñoz-Avila. LCW-based agent planning for the semantic web. In *Ontologies and the Semantic Web. Papers from the 2002 AAAI Workshop WS-02-11*, pages 63–70, 2002. AAAI Press.

[16] I. Horrocks, L. Li, D. Turi, and S. Bechhofer. The instance store: DL reasoning with large numbers of individuals. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, pages 31–40, 2004.

[17] I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *Proc. of the 2003 Int. Semantic Web Conf. (ISWC 2003)*, 2003.

[18] 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. Member submission 21 may 2004, W3C, 2004.

[19] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[20] I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ(D) description logic. In *Proc. of the Seventeenth Int. Joint Conf. on Artificial Intelligence (IJCAI2001)*, 2001.

[21] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264, May 2000.

[22] U. Hustadt, B. Motik, and U. Sattler. Reducing SHIQ⁻ description logic to disjunctive logic programs. In *Proc. of the 9th Int. Conf. on Knowledge Representation and Reasoning KR2004*.

[23] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *JACM*, 42(4):741–843, 1995.

[24] A. Y. Levy and M.-C. Rousset. Combining horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165 – 209, 1998.

[25] N. Noy, P. Hayes, B. McBride, A. Rector, and B. Vatant. Representing classes as property values on the semantic web. Working draft, W3C, 2004. http://www.w3.org/TR/swbp-classes-as-values.

[26] J. Z. Pan and I. Horrocks. OWL-E: Extending OWL with expressive datatype expressions. IMG Technical Report, Victoria University of Manchester, 2004. http://dl-web.man.ac.uk/Doc/IMGTR-OWL-E.pdf.

[27] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax. Recommendation 10 February 2004, W3C, 2004.

[28] R. Ramakrishnan. Magic templates: A spellbinding approach to logic programs. *Journal of Logic Programming*, 11(3-4):189–216, 1991.

[29] G. Schreiber. The web is not well-formed. *IEEE Intelligent Systems*, 17(2), 2002. Contribution to the section Trends and Controversies: Ontologies KISSES in Standardization.

[30] D. Tsarkov and I. Horrocks. Efficient reasoning with range and domain constraints. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, pages 41–50, 2004.

[31] R. Volz. *Web Ontology Reasoning with Logic Databases*. PhD thesis, AIFB, Karlsruhe, 2004.

[32] G. Yang, M. Kifer, and C. Zhao. FLORA-2: A rule-based knowledge representation and inference infrastructure for the semantic web. In *Proc. of the Second Int. Conf. on Ontologies, Databases and Applications of Semantics (ODBASE)*, 2003.