Reducing OWL Entailment to Description Logic Satisfiability[⋆]

Ian Horrocks[1] and Peter F. Patel-Schneider[2]

[1] Department of Computer Science
University of Manchester
Email: horrocks@cs.man.ac.uk
[2] Bell Labs Research
Lucent Technologies
Email: pfps@research.bell-labs.com

Abstract. We show how to reduce ontology entailment for the OWL DL and OWL Lite ontology languages to knowledge base satisfiability in (respectively) the $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$ description logics. This is done by first establishing a correspondence between OWL ontologies and description logic knowledge bases and then by showing how knowledge base entailment can be reduced to knowledge base satisfiability.

1 Introduction

The aim of the Semantic Web is to make web resources (not just HTML pages, but a wide range of web accessible data and services) more readily accessible to automated processes. This is to be done by augmenting existing *presentation* markup with *semantic* markup, i.e., meta-data annotations that describe their content [2]. According to widely known proposals for a Semantic Web architecture, ontologies will play a key role as they will be used as a source of shared and precisely defined terms that can be used in such metadata [15].

The importance of ontologies in semantic markup has prompted the development of several ontology languages specifically designed for this purpose. These include OIL [7], DAML+OIL [13] and OWL [4, 16]. OWL is of particular significance as it has been developed by the W3C Web Ontology working group, and is now an official W3C recommendation.

The OWL recommendation actually consists of three languages of increasing expressive power: OWL Lite, OWL DL and OWL Full. Like OWL's predecessor DAML+OIL, OWL Lite and OWL DL can be viewed as expressive description logics with an RDF syntax. They can therefore exploit the considerable existing body of description logic research. In particular, these two languages can utilize previous work reported on in the description logic literature to define their semantics and to understand their formal properties such as the decidability and complexity of key inference

[⋆] This is a revised and extended version of a paper of the same name that was presented at ISWC-2003 in October 2003.

problems [6]. OWL Full provides a more complete integration with RDF, but its formal properties are less well understood, and key inference problems would certainly be *much* harder to compute.[3] This paper, therefore, concentrates on the provision of reasoning services for OWL Lite and OWL DL, and does not consider reasoning in OWL Full.

1.1 OWL Reasoning

Reasoning with ontology languages will be important in the Semantic Web if applications are to exploit the semantics of ontology based metadata annotations. For example, if semantic search engines are to find pages based on the semantics of their annotations rather than their syntax, then they need to perform semantic reasoning in the language of the annotations. As well as providing insights into OWL's formal properties, OWL's relationship to expressive description logics provides a source of algorithms for solving key inference problems, in particular satisfiability. Moreover, in spite of the high worst case complexity of reasoning in such description logics, highly optimised implementations of these algorithms are available and have been shown to work well with realistic problems. Two difficulties arise, however, when attempting to use such implementations to provide reasoning services for OWL:

1. OWL's RDF syntax uses frame-like constructs that do not correspond directly to description logic axioms; and
2. as in RDF, OWL inference is defined in terms of ontology entailment rather than ontology satisfiability.

Note that entailment between ontologies (similarly, entailment between RDF graphs) is a basic inference task into which most other inference tasks can be transformed: given an ontology $\mathcal{O}$, a class C is subsumed by a class D with respect to $\mathcal{O}$ just in case $\mathcal{O}$ entails the ontology $\{\texttt{SubClassOf}(C\ D)\}$, and i is an instance of C with respect to $\mathcal{O}$ just in case $\mathcal{O}$ entails the ontology $\{\texttt{Individual}(i\ \texttt{type}(C)\}$. Moreover, transforming these inference tasks into ontology (graph) satisfiability requires full negation, which is not available in either RDF or RDF Schema, and is not directly supported in OWL Lite.

The obvious solution to the first difficulty is to define a mapping that decomposes OWL frames into one or more description logic axioms. It turns out, however, that the RDF syntax used in OWL cannot be directly translated into any "standard" description logic because it allows the use of anonymous individuals in axioms asserting the types of and relationships between individuals. The obvious solution to the second difficulty is to reduce entailment to satisfiability. Doing this naively would, however, require role negation, and this is not supported in any implemented description logic reasoner.

In this paper we will show that, in spite of these difficulties, ontology entailment in OWL DL and OWL Lite can be reduced to knowledge base satisfiability in the $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$ description logics respectively. This is achieved by mapping OWL to an intermediate description logic that includes a novel axiom asserting the

[3] Inference in OWL Full is clearly undecidable as OWL Full does not include restrictions on the use of transitive properties which are required in order to maintain decidability [11].

non-emptiness of a class, and by using a more sophisticated reduction to satisfiability that both eliminates this constructor and avoids the use of role negation.

This is a significant result from both a theoretical and a practical perspective: it demonstrates that computing ontology entailment in OWL DL (respectively OWL Lite) has the same complexity as computing knowledge base satisfiability in $\mathcal{SHOIN}(\mathbf{D})$ ($\mathcal{SHIF}(\mathbf{D})$), and that description logic algorithms and implementations (such as RACER [8]) can be used to provide reasoning services for OWL Lite. The design of "practical" algorithms for $\mathcal{SHOIN}(\mathbf{D})$ is still an open problem, but one that is the subject of active investigation.

2 The OWL Web Ontology Language

As mentioned above, OWL [4, 16] is an ontology language that has recently been developed by the W3C Web Ontology Working Group. OWL is defined as an extension to RDF in the form of a vocabulary entailment [9], i.e., the syntax of OWL is the syntax of RDF and the semantics of OWL are an extension of the semantics of RDF.

OWL has many features in common with description logics, but also has some significant differences. The first difference between OWL and description logics is that the syntax of OWL is the syntax of RDF. OWL information is thus encoded in RDF/XML documents [1] and parsed into RDF Graphs [14] composed of triples. Because RDF Graphs are such an impoverished syntax, many description logic constructs in OWL are encoded into several triples. Because RDF Graphs are graphs, however, it is possible to create circular syntactic structures in OWL, which are not possible in description logics. Subtle interactions between OWL and RDF cause problems with some of these circular syntactic structures.

The second difference between OWL and description logics is that OWL contains features that do not fit within the description logic framework. For example, OWL classes are objects in the domain of discourse and can be made instances of other concepts, including themselves. These two features, also present in RDF, make a semantic treatment of OWL quite different from the semantic treatment of description logics.

2.1 OWL DL and OWL Lite

Fortunately for our purpose, there are officially-defined subsets of OWL that are much closer to description logics. The larger of these subsets, called OWL DL, restricts OWL in two ways. First, unusual syntactic constructs, such as descriptions with syntactic cycles in them, are not allowed in OWL DL. Second, classes, properties, and individuals (usually called concepts, roles and individuals in description logics) must be disjoint in the semantics for OWL DL. These two restrictions make OWL DL much closer to a description logic.

Because of the syntactic restrictions in OWL DL, it is possible to develop an abstract syntax for OWL DL [16] that looks much like an abstract syntax for a powerful frame language, and is not very different from description logic syntaxes. This is very similar to the approach taken in the OIL language [7]. The abstract syntax for OWL DL has classes and data ranges, which are analogues of concepts and concrete datatypes in

Classes
A
intersectionOf($C_1 \ldots C_n$)
unionOf($C_1 \ldots C_n$)
complementOf(C)
oneOf($o_1 \ldots o_n$)
restriction(R
$\{$allValuesFrom(C)$\}$ $\{$someValuesFrom(C)$\}$
$\{$value(o)$\}$ [minCardinality(n)]
[maxCardinality(m)] [cardinality($\ell$)])
restriction(T
$\{$allValuesFrom(D)$\}$ $\{$someValuesFrom(D)$\}$
$\{$value(v)$\}$ [minCardinality(n)]
[maxCardinality(m)] [cardinality($\ell$)])
Data Ranges
B
oneOf($v_1 \ldots v_n$)

Fig. 1. OWL DL Description Constructors

description logics, and axioms and facts, which are analogues of axioms in description logics. Axioms and facts are grouped into ontologies, the analogue of description logic knowledge bases, which are the highest level of OWL DL syntax. Ontologies can input other ontologies in OWL, but this importing should be handled outside of the semantics for OWL and thus does not affect the reduction to description logics.

The constructors used to form OWL DL descriptions and data ranges are provided in Figure 1; in the figure A is a class name, C (possibly subscripted) is a class, o (possibly subscripted) is an individual name, R (possibly subscripted) is an object property (also called abstract or individual-valued properties), T (possibly subscripted) is a datatype property,[4] B is a datatype, D (possibly subscripted) is a data range, v (possibly subscripted) is a data value, and ℓ, m, n are non-negative integers. A data value is either of the form "$\ell$"`^^d`, where d is the name of a supported datatype and $\ell$ is a lexical form in that datatype, or an untyped string with an optional language tag. For example, `"1"^^xsd:integer` denotes the integer 1, whereas both `"1"^^xsd:string` and `"1"` denote one-character strings. An OWL DL or OWL Lite reasoner may support many datatypes, but *must* support at least the XML Schema datatypes xsd:integer and xsd:string. Data values of the form `"$\ell$"^^d`, where d is not a supported datatype, are also allowed in OWL DL and OWL Lite. The denotation of these data values are unconstrained.

Elements enclosed in braces (i.e., $\{$element$\}$) can be repeated zero or more times and elements enclosed in square brackets (i.e., [element]) are optional. A more leisurely description of these constructors can be found in the OWL documentation [4, 16].

[4] An object property is one that associates pairs of individuals; a datatype property associates an individual with a data value.

Class Axioms
Class(A partial $C_1 \ldots C_n$)
Class(A complete $C_1 \ldots C_n$)
EnumeratedClass(A $o_1 \ldots o_n$)
DisjointClasses($C_1 \ldots C_n$)
EquivalentClasses($C_1 \ldots C_n$)
SubClassOf(C_1 C_2)
Property Axioms
DatatypeProperty(T super(T_1) . . . super(T_n) [Functional] domain(C_1) . . . domain(C_m) range(D_1) . . . range(D_ℓ))
ObjectProperty(R super(R_1) . . . super(R_n) [inverseOf(R_0)] [Functional] [InverseFunctional] [Symmetric] [Transitive] domain(C_1) . . . domain(C_m) range(D_1) . . . range(D_ℓ))
EquivalentProperties($T_1 \ldots T_n$)
SubPropertyOf(T_1 T_2)
EquivalentProperties($R_1 \ldots R_n$)
SubPropertyOf(R_1 R_2)
Facts
Individual([o] type(C_1) . . . type(C_m) value(p_1 x_1) . . . value(p_n x_n))
SameIndividual($o_1 \ldots o_n$)
DifferentIndividuals($o_1 \ldots o_n$)

Fig. 2. OWL DL Axioms and Facts

Names in OWL are officially URI references, but all that matters here is that they are treated in our semantics as atomic names.

Classes and data ranges can be used in OWL DL axioms and facts to provide information about classes, properties, and individuals. Figure 2 provides the syntax of these axioms and facts. In this figure, the same conventions are used as in Figure 1 with the addition that value(p_i x_i) is a value condition where p_i is either a datatype property, in which case x_i is a data value, or an object property, in which case x_i is either an individual name or an individual fact.

To preserve decidability of reasoning in OWL DL, *complex* object properties cannot be specified to be transitive. An object property is complex if either

1. it is specified as being functional or inverse-functional,
2. there is some cardinality restriction that uses it,
3. it has an inverse that is complex, or
4. it has a super-property that is complex.

Again, a more leisurely description of these constructors can be found in the OWL documentation [4, 16]. Figure 2 ignores annotations and deprecation, which allow uninterpreted information to be associated with classes and properties, but which are not interesting from a logical point of view.

Because of the syntactic restrictions in OWL DL, metaclasses and other notions that do not fit into the description logic semantic framework can be ignored. In fact, OWL

DL has a semantics that is very much in the description logic style, and that has been shown to be equivalent to the RDF-style semantics for all of OWL [16]. The semantics for OWL DL will be presented below.

There is a subset of OWL DL, called OWL Lite, the motivation for which is increased ease of implementation. This is achieved by supporting fewer constructors than OWL DL, and by limiting the use of some of these constructors. In particular, OWL Lite does not support the `oneOf` constructor (equivalent to description logic *nominals*), as this constructor is known to increase theoretical complexity and to lead to difficulties in the design of practical algorithms [10]. In Section 5 we will examine the differences between OWL DL and OWL Lite in more detail, and explore their impact on the reduction from OWL entailment to description logic satisfiability.

2.2 Semantics for OWL DL

OWL DL has two forms of semantic specification: a direct model-theoretic semantics, and an RDF-compatible model-theoretic semantics [16]. The two are said to have "a strong correspondence", but the specification explicitly states that the direct model-theoretic semantics takes precedence. We will, therefore, only consider the direct model-theoretic semantics, and from now on when we refer to the semantics for OWL DL (or OWL Lite), this can be taken to mean the direct model-theoretic semantics.

The semantics for OWL DL is fairly standard by description logic standards. The OWL semantic domain is a set whose elements can be disjointly divided into abstract objects (the abstract domain, written $\Delta^{\mathcal{I}}$) and datatype values (the datatype or concrete domain, written $\Delta^{\mathcal{I}}_{\mathbf{D}}$ and often called concrete objects). Datatypes in OWL are derived from the built-in XML Schema datatypes [3], with inappropriate datatypes removed, although as mentioned in Section 2.1, an OWL DL or OWL Lite reasoner may not support all of these datatypes. Datatype values are denoted by special literal constructs in the syntax, as indicated above.

In order to be closer to the RDF semantics [9], an interpretation in the semantics for OWL DL is officially a sextuple consisting of the abstract domain, the concrete domain, a mapping from class names into subsets of the abstract domain and from datatype names into subsets of the concrete domain, a mapping from object properties to sets of pairs over the abstract domain and from datatype properties to sets of pairs from the abstract domain and the concrete domain,[5] a mapping from individual names to values in the abstract domain, and a mapping from literals to values in the concrete domain. This does not quite match up with the description logic method of using a two-tuple consisting of the domain (written $\Delta^{\mathcal{I}}$) and a single mapping (written $\cdot^{\mathcal{I}}$) for concepts, properties, and individuals, with datatypes handled as an external parameter. There is, however, an obvious isomorphism between the two methods, and so either one can be used for our purposes.

In OWL DL all classes are interpreted as subsets of the abstract domain, and for each constructor the semantics of the resulting class is defined in terms of the semantics

[5] This mapping is also used to provide meaning for annotations, which are not considered in this paper.

of its components. For example, given two classes C_1 and C_2, the interpretation of the intersection of C_1 and C_2 is defined to be the intersection of the interpretations of C_1 and C_2 (i.e., ${C_1}^{\mathcal{I}} \cap {C_2}^{\mathcal{I}}$).

OWL DL axioms and facts result in semantic conditions on interpretations. For example, an axiom asserting that D_1 is a subclass of D_2 results in the semantic condition that the interpretation of D_1 must be a subset of the interpretation of D_2, (written ${D_1}^{\mathcal{I}} \subset {D_2}^{\mathcal{I}}$), while a fact asserting that o has type D results in the semantic condition that the interpretation of o must be an element of the set that is the interpretation of D (written $o^{\mathcal{I}} \in D^{\mathcal{I}}$), just as happens in description logic semantics. An OWL DL ontology O is satisfied by an interpretation $\mathcal{I}$ just when all of the semantic conditions resulting from the axioms and facts in $\mathcal{O}$ are satisfied by $\mathcal{I}$, just as is the case in description logic knowledge bases. Because this part of semantics for OWL DL is so close to the semantics of description logics, it will not be further provided here; instead, we will use the description logic semantics directly, as we will mainly be interested in description logic knowledge bases derived from OWL ontologies. More details of OWL DL semantics can be found in the OWL documentation [16].

The main semantic relationship in OWL DL is entailment—a relationship between pairs of OWL ontologies. An ontology O_1 entails an ontology O_2, written $O_1 \models O_2$, exactly when all interpretations that satisfy O_1 also satisfy O_2. This semantic relationship is different from the standard description logic relationships, such as knowledge base and concept satisfiability. The main goal of this paper is to show how OWL DL entailment can be transformed into DL knowledge base (un)satisfiability, and that the two problems have the same complexity.

3 $\mathcal{SHOIN}$(D) and $\mathcal{SHIF}$(D)

The main description logic that we will be using in this paper is $\mathcal{SHOIN}(\mathbf{D})$, which is similar to the well known $\mathcal{SHOQ}(\mathbf{D})$ description logic [10], but is extended with inverse roles ($\mathcal{I}$) and restricted to unqualified number restrictions ($\mathcal{N}$).

In description logics, a datatype theory **D** is a mapping from a set of datatypes to a set of values, e.g., from xsd:integer to the integers, plus a mapping from data values to their denotation which must be one of the set of values, e.g., from `"1"^^xsd:integer` to the integer 1. The datatype (or concrete) domain, written $\Delta^{\mathcal{I}}_{\mathbf{D}}$, is the union of the mappings of the datatypes.

Given a datatype theory **D**, let **A**, $\mathbf{R}_A$, $\mathbf{R}_{\mathbf{D}}$, and **I** be pairwise disjoint sets of *concept names*, *abstract role names*, *datatype (or concrete) role names*, and *individual names*.[6] The set of $\mathcal{SHOIN}(\mathbf{D})$-roles is $\mathbf{R}_A \cup \{R^- \mid R \in \mathbf{R}_A\} \cup \mathbf{R}_{\mathbf{D}}$. In order to avoid considering roles such as R^{--} we will define $\mathsf{Inv}(R)$ s.t. $\mathsf{Inv}(R) = R^-$ and $\mathsf{Inv}(R^-) = R$. The set of $\mathcal{SHOIN}(\mathbf{D})$-concepts is the smallest set that can be built using the constructors in Figure 3.

Figure 3 also gives the axiom syntax for $\mathcal{SHOIN}^+(\mathbf{D})$, an extension of $\mathcal{SHOIN}(\mathbf{D})$ with the *concept existence* axiom (the last axiom in Figure 3), which

[6] Datatype roles names are generally referred to as concrete role names in the description logic literature.

Constructor Name	**Syntax**	**Semantics**
atomic concept **A**	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
datatype **D**	D	$D^{\mathbf{D}} \subseteq \Delta^{\mathcal{I}}_{\mathbf{D}}$
abstract role $\mathbf{R}_A$	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
datatype role $\mathbf{R}_{\mathbf{D}}$	U	$U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}_{\mathbf{D}}$
individuals **I**	o	$o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
data values	v	$v^{\mathcal{I}} = v^{\mathbf{D}}$
inverse role	R^-	$(R^-)^{\mathcal{I}} = (R^{\mathcal{I}})^-$
top	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
bottom	$\bot$	$\bot^{\mathcal{I}} = \{\}$
conjunction	$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
disjunction	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
negation	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
oneOf	$\{o_1, \ldots, o_n\}$	$\{o_1, \ldots, o_n\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}}\}$
exists restriction	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y. \langle x, y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
value restriction	$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y. \langle x, y\rangle \in R^{\mathcal{I}} \to y \in C^{\mathcal{I}}\}$
atleast restriction	$\geqslant n\, R$	$(\geqslant n\, R)^{\mathcal{I}} = \{x \mid \sharp(\{y. \langle x, y\rangle \in R^{\mathcal{I}}\}) \geqslant n\}$
atmost restriction	$\leqslant n\, R$	$(\leqslant n\, R)^{\mathcal{I}} = \{x \mid \sharp(\{y. \langle x, y\rangle \in R^{\mathcal{I}}\}) \leqslant n\}$
datatype exists	$\exists U.D$	$(\exists U.D)^{\mathcal{I}} = \{x \mid \exists y. \langle x, y\rangle \in U^{\mathcal{I}} \text{ and } y \in D^{\mathbf{D}}\}$
datatype value	$\forall U.D$	$(\forall U.D)^{\mathcal{I}} = \{x \mid \forall y. \langle x, y\rangle \in U^{\mathcal{I}} \to y \in D^{\mathbf{D}}\}$
datatype atleast	$\geqslant n\, U$	$(\geqslant n\, U)^{\mathcal{I}} = \{x \mid \sharp(\{y. \langle x, y\rangle \in U^{\mathcal{I}}\}) \geqslant n\}$
datatype atmost	$\leqslant n\, U$	$(\leqslant n\, U)^{\mathcal{I}} = \{x \mid \sharp(\{y. \langle x, y\rangle \in U^{\mathcal{I}}\}) \leqslant n\}$
datatype oneOf	$\{v_1, \ldots\}$	$\{v_1, \ldots\}^{\mathcal{I}} = \{v_1^{\mathcal{I}}, \ldots\}$
Axiom Name	**Syntax**	**Semantics**
concept inclusion	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
object role inclusion	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
object role transitivity	$\mathsf{Trans}(R)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$
datatype role inclusion	$U_1 \sqsubseteq U_2$	$U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}}$
individual inclusion	$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
individual equality	$a = b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
individual inequality	$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$
concept existence	$\exists C$	$\sharp(C^{\mathcal{I}}) \geqslant 1$

Fig. 3. Syntax and semantics of $\mathcal{SHOIN}^+(\mathbf{D})$

is used internally in our translation. Concept existence axioms will be eliminated in the final step of our translation, leaving only $\mathcal{SHOIN}(\mathbf{D})$ axioms.

A $\mathcal{SHOIN}^+(\mathbf{D})$ *knowledge base* $\mathcal{K}$ is a finite set of $\mathcal{SHOIN}^+(\mathbf{D})$ axioms. We will use $\underline{\boxast}$ to denote the transitive reflexive closure of $\sqsubseteq$ on roles, i.e., for two roles S, R in $\mathcal{K}$, $S \,\underline{\boxast}\, R$ in $\mathcal{K}$ if $S = R$, $S \sqsubseteq R \in \mathcal{K}$, $\mathsf{Inv}(S) \sqsubseteq \mathsf{Inv}(R) \in \mathcal{K}$, or there exists some role Q such that $S \sqsubseteq^* Q$ in $\mathcal{K}$ and $Q \sqsubseteq^* R$ in $\mathcal{K}$. A role R is called *simple* in $\mathcal{K}$ if for each role S s.t. $S \sqsubseteq^* R$ in $\mathcal{K}$, $\mathsf{Trans}(S) \notin \mathcal{K}$ and $\mathsf{Trans}(\mathsf{Inv}(S)) \notin \mathcal{K}$. To maintain decidability, a knowledge base must have no number restrictions on non-simple roles [11].

The semantics of $\mathcal{SHOIN}^+(\mathbf{D})$ is given by means of an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consisting of a non-empty domain $\Delta^\mathcal{I}$, disjoint from the datatype domain $\Delta^\mathcal{I}_\mathbf{D}$, and a mapping $\cdot^\mathcal{I}$, which interprets atomic and complex concepts, roles, and nominals according to Figure 3. (In Figure 3, $\sharp$ is set cardinality.)

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ *satisfies* a $\mathcal{SHOIN}^+(\mathbf{D})$-axiom under the conditions given in Figure 3. An interpretation satisfies a knowledge base $\mathcal{K}$ iff it satisfies each axiom in $\mathcal{K}$; we will often call such an interpretation a *model* of $\mathcal{K}$. A knowledge base $\mathcal{K}$ is *satisfiable* (*unsatisfiable*) iff there exists (does not exist) a model of $\mathcal{K}$. A $\mathcal{SHOIN}^+(\mathbf{D})$-concept C is satisfiable with respect to a knowledge base $\mathcal{K}$ iff there is a model $\mathcal{I}$ of $\mathcal{K}$ with $C^\mathcal{I} \neq \emptyset$. A concept C is *subsumed* by a concept D with respect to $\mathcal{K}$ iff $C^\mathcal{I} \sqsubseteq D^\mathcal{I}$ in every model $\mathcal{I}$ of $\mathcal{K}$. Two concepts are said to be equivalent with respect to $\mathcal{K}$ iff they subsume each other with respect to $\mathcal{K}$. A knowledge base $\mathcal{K}_1$ entails a knowledge base $\mathcal{K}_2$ iff every model of $\mathcal{K}_1$ is also a model of $\mathcal{K}_2$.

Although this is not usually done in description logics, we define a notion of entailment in $\mathcal{SHOIN}^+(\mathbf{D})$ in the same way as it was defined for OWL DL. One $\mathcal{SHOIN}^+(\mathbf{D})$ knowledge base entails another, written $\mathcal{K} \models \mathcal{K}'$, if every model of the first knowledge base, $\mathcal{K}$ is also a model of the second, $\mathcal{K}'$. It is easy to show that $\mathcal{K} \models \mathcal{K}'$ iff $\mathcal{K} \models A$ for every axiom A in $\mathcal{K}'$.

The description logic $\mathcal{SHIF}(\mathbf{D})$ is just $\mathcal{SHOIN}(\mathbf{D})$ without the oneOf constructor and with the atleast and atmost constructors limited to 0 and 1. We define $\mathcal{SHIF}^+(\mathbf{D})$ as $\mathcal{SHIF}(\mathbf{D})$ extended with the concept existence axiom.

4 From OWL DL Entailment to $\mathcal{SHOIN}(\mathbf{D})$ Unsatisfiability

We will now show how to translate OWL DL entailment into $\mathcal{SHOIN}(\mathbf{D})$ unsatisfiability. The first step of our process is to translate an entailment between OWL DL ontologies into an entailment between knowledge bases in $\mathcal{SHOIN}^+(\mathbf{D})$. Then $\mathcal{SHOIN}^+(\mathbf{D})$ entailment is transformed into unsatisfiability of $\mathcal{SHOIN}(\mathbf{D})$ knowledge bases. Note that concept existence axioms are eliminated in this last step, leaving a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base.

From now on **D** will be a particular kind of datatype theory, namely for the well-behaved XML Schema datatypes [3] plus a datatype for untyped OWL literals plus one other datatype, whose extension is the entire datatype domain, and using the OWL syntax for data values. (See the OWL documentation [16] for the particulars of which XML Schema datatypes are well-behaved and why.) The extra datatype, which cannot occur in the ontologies being translated, will be used as a way to write unknown data values.

It is easy to see that these datatypes comprise a datatype theory.

OWL fragment F	Translation $\mathcal{V}(F)$
A, OWL class name	A
B, OWL datatype name	B
R, OWL object property name	R
T, OWL datatype property name	T
o, OWL individual name	o
v, OWL data value	v
intersectionOf($C_1 \ldots C_n$)	$\mathcal{V}(C_1) \sqcap \ldots \sqcap \mathcal{V}(C_n)$
unionOf($C_1 \ldots C_n$)	$\mathcal{V}(C_1) \sqcup \ldots \sqcup \mathcal{V}(C_n)$
complementOf(C)	$\neg\mathcal{V}(C)$
oneOf($o_1 \ldots o_n$)	$\{\mathcal{V}(o_1), \ldots, \mathcal{V}(o_n)\}$
restriction($R\ r_1\ r_2 \ldots r_n$)	$\mathcal{V}$(restriction($R\ r_1$)) $\sqcap \ldots \sqcap$ $\mathcal{V}$(restriction($R\ r_n$))
restriction(R allValuesFrom(C))	$\forall\mathcal{V}(R).\mathcal{V}(C)$
restriction(R someValuesFrom(C))	$\exists\mathcal{V}(R).\mathcal{V}(C)$
restriction(R value(o))	$\exists\mathcal{V}(R).\{\mathcal{V}(o)\}$
restriction(R minCardinality(n))	$\geqslant n\,\mathcal{V}(R)$
restriction(R maxCardinality(n))	$\leqslant n\,\mathcal{V}(R)$
restriction(R cardinality(n))	$\geqslant n\,\mathcal{V}(R) \sqcap \leqslant n\,\mathcal{V}(R)$
restriction($T\ r_1\ r_2 \ldots r_n$)	$\mathcal{V}$(restriction($T\ r_1$)) $\sqcap \ldots \sqcap$ $\mathcal{V}$(restriction($T\ r_n$))
restriction(T allValuesFrom(D))	$\forall\mathcal{V}(T).\mathcal{V}(D)$
restriction(T someValuesFrom(C))	$\exists\mathcal{V}(T).\mathcal{V}(D)$
restriction(T value(v))	$\exists\mathcal{V}(T).\{\mathcal{V}(v)\}$
restriction(T minCardinality(n))	$\geqslant n\,\mathcal{V}(T)$
restriction(T maxCardinality(n))	$\leqslant n\,\mathcal{V}(T)$
restriction(T cardinality(n))	$\geqslant n\,\mathcal{V}(T) \sqcap \leqslant n\,\mathcal{V}(T)$
oneOf($v_1 \ldots v_n$)	$\{\mathcal{V}(v_1), \ldots, \mathcal{V}(v_n)\}$

Fig. 4. Translation from OWL classes and names to $\mathcal{SHOIN}(\mathbf{D})$

4.1 From OWL DL to $\mathcal{SHOIN}^+(\mathbf{D})$

An OWL DL ontology is translated into a $\mathcal{SHOIN}^+(\mathbf{D})$ knowledge base by taking each axiom and fact in the ontology and translating it into one or more axioms in the knowledge base.

For OWL DL axioms, this translation is very natural, and is almost identical to the translation of OIL described by Decker *et al.* [5]. For example, the OWL DL axiom Class(A complete $C_1 \ldots C_n$) is translated into the pair of $\mathcal{SHOIN}^+(\mathbf{D})$ axioms $A \sqsubseteq \mathcal{V}(C_1) \sqcap \ldots \sqcap \mathcal{V}(C_n)$ and $\mathcal{V}(C_1) \sqcap \ldots \sqcap \mathcal{V}(C_n) \sqsubseteq A$, where $\mathcal{V}$ is the obvious translation from OWL classes to description logic concepts, again very similar to the transformation described by Decker *et al.* [5]. Similarly, an OWL DL axiom `DisjointClasses(`C_1`...`C_n`)` is translated into the $\mathcal{SHOIN}^+(\mathbf{D})$ axioms $\mathcal{V}(C_i) \sqsubseteq \neg\mathcal{V}(C_j)$ for $1 \leq i < j \leq n$. The translation from OWL DL classes to $\mathcal{SHOIN}(\mathbf{D})$ classes is given in Figure 4 and the translation from OWL DL axioms to $\mathcal{SHOIN}(\mathbf{D})$ axioms is given in Figure 5.

The translation of OWL DL facts to $\mathcal{SHOIN}^+(\mathbf{D})$ axioms is more complex. This is because facts can be stated with respect to anonymous individuals, and can include relationships to other (possibly anonymous) individuals. For example, the fact Individual(type(C) value(R Individual(type(D)))) states that there exists an individual

OWL fragment F	Translation $\mathcal{V}(F)$
Class(A partial $C_1 \ldots C_n$)	$A \sqsubseteq \mathcal{V}(C_1) \sqcap \ldots \sqcap \mathcal{V}(C_n)$
Class(A complete $C_1 \ldots C_n$)	$A \sqsubseteq \mathcal{V}(C_1) \sqcap \ldots \sqcap \mathcal{V}(C_n), \mathcal{V}(C_1) \sqcap \ldots \sqcap \mathcal{V}(C_n) \sqsubseteq A$
EnumeratedClass(A $o_1 \ldots o_n$)	$A \sqsubseteq \{\mathcal{V}(o_1), \ldots, \mathcal{V}(o_n)\}, \{\mathcal{V}(o_1), \ldots, \mathcal{V}(o_n)\} \sqsubseteq A$
DisjointClasses($C_1 \ldots C_n$)	$\mathcal{V}(C_i) \sqsubseteq \neg\mathcal{V}(C_j), 1 \leq i < j \leq n$
EquivalentClasses($C_1 \ldots C_n$)	$\mathcal{V}(C_i) = \mathcal{V}(C_{i+1}), 1 \leq i < n$
SubClassOf(C_1 C_2)	$\mathcal{V}(C_1) \sqsubseteq \mathcal{V}(C_2)$
DatatypeProperty(T r_1 $r_2 \ldots r_n$)	$\mathcal{V}$(DatatypeProperty(T r_1)), $\ldots$, $\mathcal{V}$(DatatypeProperty(T r_n))
DatatypeProperty(T super(T_1))	$\mathcal{V}(T) \sqsubseteq \mathcal{V}(T_1)$
DatatypeProperty(T Functional)	$\top \sqsubseteq \leqslant 1\, \mathcal{V}(T)$
DatatypeProperty(T domain(C))	$\geqslant 1\, \mathcal{V}(T) \sqsubseteq \mathcal{V}(C)$
DatatypeProperty(T range(D))	$\top \sqsubseteq \forall\mathcal{V}(T).\mathcal{V}(D)$
ObjectProperty(R r_1 $r_2 \ldots r_n$)	$\mathcal{V}$(ObjectProperty(R r_1)), $\ldots$, $\mathcal{V}$(ObjectProperty(R r_n))
ObjectProperty(R super(R_1))	$\mathcal{V}(R) \sqsubseteq \mathcal{V}(R_1)$
ObjectProperty(R inverseOf(R_0))	$\mathcal{V}(R) \sqsubseteq \mathcal{V}(R)^-$
ObjectProperty(R Functional)	$\top \sqsubseteq \leqslant 1\, \mathcal{V}(R)$
ObjectProperty(R InverseFunctional)	$\top \sqsubseteq \leqslant 1\, \mathcal{V}(R)^-$
ObjectProperty(R Symmetric)	$\mathcal{V}(R) \sqsubseteq \mathcal{V}(R)^-$
ObjectProperty(R Transitive)	$\mathsf{Trans}(\mathcal{V}(R))$
ObjectProperty(R domain(C))	$\geqslant 1\, \mathcal{V}(R) \sqsubseteq \mathcal{V}(C)$
ObjectProperty(R range(C))	$\top \sqsubseteq \forall\mathcal{V}(R).\mathcal{V}(C)$
EquivalentProperties($T_1 \ldots T_n$)	$\mathcal{V}(T_i) \sqsubseteq \mathcal{V}(T_j), 1 \leq i, j \leq n$
SubPropertyOf(T_1 T_2)	$\mathcal{V}(T_1) \sqsubseteq \mathcal{V}(T_1)$
EquivalentProperties($R_1 \ldots R_n$)	$\mathcal{V}(R_i) \sqsubseteq \mathcal{V}(R_j), 1 \leq i, j \leq n$
SubPropertyOf(R_1 R_2)	$\mathcal{V}(R_1) \sqsubseteq \mathcal{V}(R_1)$

Fig. 5. Translation from OWL axioms to $\mathcal{SHOIN}(\mathbf{D})$

that is an instance of class C and is related via the property R to an individual that is an instance of the class D, without naming either of the individuals.

The need to translate this kind of fact is the reason for introducing the $\mathcal{SHOIN}^+(\mathbf{D})$ existence axiom. For example, the above fact can be translated into the axiom $\exists(C \sqcap \exists R.D)$, which states that there exists some instance of the concept $C \sqcap \exists R.D$, i.e., an individual that is an instance of C and is related via the role R to an instance of the concept D. Figure 6 describes a translation $\mathcal{F}$ that transforms OWL Individual facts into $\mathcal{SHOIN}^+(\mathbf{D})$ existence axioms (and the other OWL facts into $\mathcal{SHOIN}(\mathbf{D})$ axioms).

Theorem 1. *The translation from OWL DL to $\mathcal{SHOIN}^+(\mathbf{D})$ preserves satisfiability. That is, an OWL DL axiom or fact is satisfied by an interpretation $\mathcal{I}$ if and only if the translation is satisfied by $\mathcal{I}$.*[7]

Proof. A simple recursive argument based on the semantics of OWL DL and $\mathcal{SHOIN}^+(\mathbf{D})$ shows that the extension of OWL DL classes, data ranges, and pieces

[7] The statement of the theorem here ignores the minor differences between OWL DL interpretations and $\mathcal{SHOIN}^+(\mathbf{D})$ interpretations. A stricter account would have to worry about these stylistic differences.

OWL fragment F	Translation $\mathcal{F}(F)$
Individual($x_1 \ldots x_n$)	$\exists(\mathcal{F}(x_1) \sqcap \ldots \sqcap \mathcal{F}(x_n))$
type(C)	$\mathcal{V}(C)$
value($R\ x$)	$\exists R.\mathcal{F}(x)$
value($T\ v$)	$\exists T.\{\mathcal{V}(v)\}$
o	$\{\mathcal{V}(o)\}$
SameIndividual($o_1 \ldots o_n$)	$\mathcal{V}(o_i) = \mathcal{V}(o_j), 1 \leq i < j \leq n$
DifferentIndividuals($o_1 \ldots o_n$)	$\mathcal{V}(o_i) \neq \mathcal{V}(o_j), 1 \leq i < j \leq n$

Fig. 6. Translation from OWL facts to $\mathcal{SHOIN}^+(\mathbf{D})$

Axiom A	Transformation $\mathcal{G}(A)$
$c \sqsubseteq d$	$x : c \sqcap \neg d$
$\exists c$	$\top \sqsubseteq \neg c$
$\mathsf{Trans}(r)$	$x : \exists r.\exists r.\{y\} \sqcap \neg\exists r.\{y\}$
$r \sqsubseteq s$	$x : \exists r.\{y\} \sqcap \neg\exists s.\{y\}$
$f \sqsubseteq g$	$x : \exists f.\{b\} \sqcap \neg\exists g.\{b\}$ for b a fresh data value of the extra datatype
$a = b$	$a \neq b$
$a \neq b$	$a = b$

Fig. 7. Translation from Entailment to Unsatisfiability

of Individual facts is maintained in the translation. Similarly, a simple semantics based argument shows that the translation of OWL DL axioms and facts preserves satisfaction.

The above translation increases the size of an ontology to at most the square of its size. It can easily be performed in time linear in the size of the resultant knowledge base.

4.2 From Entailment to Unsatisfiability

The next step of our process is to transform $\mathcal{SHOIN}^+(\mathbf{D})$ knowledge base entailment to $\mathcal{SHOIN}(\mathbf{D})$ knowledge base unsatisfiability. We do this to relate our new notion of description logic entailment to the well-known operation of description logic knowledge base unsatisfiability.

We recall from Section 3 that $\mathcal{K} \models \mathcal{K}'$ iff $\mathcal{K} \models A$ for every axiom A in $\mathcal{K}'$. We therefore define (in Figure 7) a translation, $\mathcal{G}$, such that $\mathcal{K} \models A$ iff $\mathcal{K} \cup \{\mathcal{G}(A)\}$ is unsatisfiable, for $\mathcal{K}$ a $\mathcal{SHOIN}^+(\mathbf{D})$ knowledge base and A a $\mathcal{SHOIN}^+(\mathbf{D})$ axiom. In this transformation we have need of names of various sorts that do not occur in the knowledge base or axiom; following standard practice we will call these fresh names. Throughout the translation, x and y are fresh individual names.

Most of the translations in $\mathcal{G}$ are quite standard and simple. For example, an object role inclusion axiom $r \sqsubseteq s$ is translated into an axiom $x : \exists r.\{y\} \sqcap \neg\exists s.\{y\}$ that requires the existence of an individual that is related to some other individual by r but not by s; a knowledge base $\mathcal{K} \cup \{x : \exists r.\{y\} \sqcap \neg\exists s.\{y\}\}$ will clearly be unsatisfiable iff

$\mathcal{K} \models r \sqsubseteq s$. The only unusual translation is for datatype role inclusions $f \sqsubseteq g$. We have included an extra datatype, whose semantics are purposely left underdefined, precisely to serve as a source of fresh values whose denotation can be arbitrarily adjusted.

The translation $\mathcal{G}$ increases the size of an axiom by at most a constant amount. It can easily be performed in time linear in the size of the axiom.

The translation $\mathcal{G}$ eliminates concept existence axioms from the knowledge base $\mathcal{K}'$ on the right-hand side of the entailment. Our last step is to eliminate concept existence axioms from the knowledge base $\mathcal{K}$ on the left-hand side of the entailment. We do this by applying a translation $\mathcal{E}(\mathcal{K})$ that replaces each axiom of the form $\exists C \in \mathcal{K}$ with an axiom $a : C$, for a a fresh individual name. It is obvious that this translation preserves satisfiability, can be easily performed, and only increases the size of a knowledge base by a linear amount.

Theorem 2. *Let $\mathcal{K}$ and $\mathcal{K}'$ be $\mathcal{SHOIN}^+(\mathbf{D})$ knowledge bases. Then $\mathcal{K} \models \mathcal{K}'$ iff the $\mathcal{SHOIN}(\mathbf{D})$ knowledge base $\mathcal{E}(\mathcal{K}) \cup \{\mathcal{G}(\mathcal{A})\}$ is unsatisfiable for every axiom A in $\mathcal{K}'$.*

Proof. *Firstly, $\mathcal{K} \models \mathcal{K}'$ iff $\mathcal{E}(\mathcal{K}) \models \mathcal{K}'$. This follows from the obvious correspondence between models of $\mathcal{K}$ and models of $\mathcal{E}(\mathcal{K})$: a model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$ is also a model of $\mathcal{K}$, because for every axiom of the form $\exists C \in \mathcal{K}$ there is an axiom $a : C \in \mathcal{E}(\mathcal{K})$, so $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\#(C^{\mathcal{I}}) \geqslant 1$; a model $\mathcal{I}$ of $\mathcal{K}$ can be trivially extended to a model of $\mathcal{E}(\mathcal{K})$ by interpreting each fresh individual a in an axiom $a : C$ in $\mathcal{E}(\mathcal{K})$ as an element of $C^{\mathcal{I}}$ (such an element must exist as there is a corresponding axiom $\exists C$ in $\mathcal{K}$).*

Given that $\mathcal{E}(\mathcal{K}) \models \mathcal{K}'$ iff $\mathcal{E}(\mathcal{K}) \models \mathcal{A}$ for every axiom A in $\mathcal{K}'$, we only need to show that $\mathcal{E}(\mathcal{K}) \models \mathcal{A}$ iff $\mathcal{E}(\mathcal{K}) \cup \{\mathcal{G}(\mathcal{A})\}$ is unsatisfiable for any given axiom A. We can do this on a case by case basis for the seven kinds of axiom described in Figure 7. In most cases the proof is a trivial consequence of the semantics, and of the fact that the fresh individuals introduced by the transformation can be interpreted as any element of $\Delta^{\mathcal{I}}$ (because they are not mentioned elsewhere in $\mathcal{E}(\mathcal{K})$). In the following, c, d are concepts, r, s are roles, a, b are individuals, d is a data value, x, y are fresh individuals, v, w, z are elements of $\Delta^{\mathcal{I}}$ and i is an element of $\Delta^{\mathcal{I}}_{\mathbf{D}}$. We will often refer to an extension *of an interpretation $\mathcal{I}$, meaning an interpretation $\mathcal{I}'$ in which $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ and $\cdot^{\mathcal{I}'}$ is extended to interpret fresh individuals.*

- *$\mathcal{E}(\mathcal{K}) \models c \sqsubseteq d$ iff $\mathcal{E}(\mathcal{K}) \cup \{x : c \sqcap \neg d\}$ is not satisfiable. If $\mathcal{E}(\mathcal{K}) \models c \sqsubseteq d$ then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $c^{\mathcal{I}} \subseteq d^{\mathcal{I}}$ and $(c \sqcap \neg d)^{\mathcal{I}} = \emptyset$, so $\mathcal{I}$ cannot satisfy $x : c \sqcap \neg d$. For the converse, if $\mathcal{I}$ is a model of $\mathcal{E}(\mathcal{K})$ in which $c^{\mathcal{I}} \nsubseteq d^{\mathcal{I}}$, then there exists some $w \in (c \sqcap \neg d)^{\mathcal{I}}$, and $\mathcal{I}$ can be extended to $\mathcal{I}'$ such that $x^{\mathcal{I}'} = w$. $\mathcal{I}'$ therefore satisfies $x : c \sqcap \neg d$, and it is still a model of $\mathcal{E}(\mathcal{K})$ because x is not mentioned in $\mathcal{E}(\mathcal{K})$, so $\mathcal{I}'$ is a model of $\mathcal{E}(\mathcal{K}) \cup \{x : c \sqcap \neg d\}$.*
- *$\mathcal{E}(\mathcal{K}) \models \exists c$ iff $\mathcal{E}(\mathcal{K}) \cup \{\top \sqsubseteq \neg c\}$ is not satisfiable. If $\mathcal{E}(\mathcal{K}) \models \exists c$, then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $c^{\mathcal{I}} \neq \emptyset$, so $(\neg c)^{\mathcal{I}} \subset \Delta^{\mathcal{I}}$ and $\mathcal{I}$ does not satisfy $\top \sqsubseteq \neg c$. For the converse, if $\mathcal{E}(\mathcal{K}) \cup \{\top \sqsubseteq \neg c\}$ is not satisfiable, then $(\neg c)^{\mathcal{I}} \subset \Delta^{\mathcal{I}}$ and $c^{\mathcal{I}} \neq \emptyset$ in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$.*
- *$\mathcal{E}(\mathcal{K}) \models \mathsf{Trans}(r)$ iff $\mathcal{E}(\mathcal{K}) \cup \{x : \exists r.\exists r.\{y\} \sqcap \neg \exists r.\{y\}\}$ is not satisfiable. If $\mathcal{E}(\mathcal{K}) \models \mathsf{Trans}(r)$, then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $r^{\mathcal{I}} = (r^{\mathcal{I}})^+$, and $\{(x^{\mathcal{I}}, w), (w, y^{\mathcal{I}})\} \subseteq r^{\mathcal{I}}$ implies $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$, so $\mathcal{I}$ cannot satisfy $x : \exists r.\exists r.\{y\} \sqcap \neg \exists r.\{y\}$. For the converse, if $\mathcal{I}$ is a model of $\mathcal{E}(\mathcal{K})$ in which in which for some v, w, z, $\{(v, w), (w, z)\} \subseteq$*

$r^{\mathcal{I}}$ but $(v,z) \notin r^{\mathcal{I}}$, then $\mathcal{I}$ can be extended to $\mathcal{I}'$ such that $x^{\mathcal{I}'} = v$ and $y^{\mathcal{I}'} = z$, so $x^{\mathcal{I}'} \in (\exists r.\exists r.\{y\})^{\mathcal{I}'}$, $x^{\mathcal{I}'} \in (\neg\exists r.\{y\})^{\mathcal{I}'}$ and $x^{\mathcal{I}'} \in (\exists r.\exists r.\{y\} \sqcap \neg\exists r.\{y\})^{\mathcal{I}'}$. $\mathcal{I}'$ therefore satisfies $x : \exists r.\exists r.\{y\} \sqcap \neg\exists r.\{y\}$, and it is still a model of $\mathcal{E}(\mathcal{K})$ because x is not mentioned in $\mathcal{E}(\mathcal{K})$, so $\mathcal{I}'$ is a model of $\mathcal{E}(\mathcal{K}) \cup \{x : \exists r.\exists r.\{y\} \sqcap \neg\exists r.\{y\}\}$.

– *$\mathcal{E}(\mathcal{K}) \models r \sqsubseteq s$ iff $\mathcal{E}(\mathcal{K}) \cup \{x : \exists r.\{y\} \sqcap \neg\exists s.\{y\}\}$ is not satisfiable. If $\mathcal{E}(\mathcal{K}) \models r \sqsubseteq s$, then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$ implies $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in s^{\mathcal{I}}$, so $\mathcal{I}$ cannot satisfy $x : \exists r.\{y\} \sqcap \neg\exists s.\{y\}$. For the converse, if $\mathcal{I}$ is a model of $\mathcal{E}(\mathcal{K})$ in which for some v, w, $(v,w) \in r^{\mathcal{I}}$ but $(v,w) \notin s^{\mathcal{I}}$, then $\mathcal{I}$ can be extended to $\mathcal{I}'$ such that $x^{\mathcal{I}'} = v$ and $y^{\mathcal{I}'} = w$, so $x^{\mathcal{I}'} \in (\exists r.\{y\})^{\mathcal{I}'}$, $x^{\mathcal{I}'} \in (\neg\exists s.\{y\})^{\mathcal{I}'}$ and $x^{\mathcal{I}'} \in (\exists r.\{y\} \sqcap \neg\exists s.\{y\})^{\mathcal{I}'}$. $\mathcal{I}'$ therefore satisfies $x : \exists r.\{y\} \sqcap \neg\exists s.\{y\}$, and it is still a model of $\mathcal{E}(\mathcal{K})$ because neither x nor y is mentioned in $\mathcal{E}(\mathcal{K})$, so $\mathcal{I}'$ is a model of $\mathcal{E}(\mathcal{K}) \cup \{x : \exists r.\{y\} \sqcap \neg\exists s.\{y\}\}$.*

– *$\mathcal{E}(\mathcal{K}) \models f \sqsubseteq g$ iff $\mathcal{E}(\mathcal{K}) \cup \{x : \exists f.\{d\} \sqcap \neg\exists g.\{d\}\}$ is not satisfiable, where d is a fresh data value of the extra datatype (i.e., a data value not mentioned in $\mathcal{E}(\mathcal{K})$). If $\mathcal{E}(\mathcal{K}) \models f \sqsubseteq g$, then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $(x^{\mathcal{I}}, d^{\mathcal{I}}) \in f^{\mathcal{I}}$ implies $(x^{\mathcal{I}}, d^{\mathcal{I}}) \in g^{\mathcal{I}}$, so $(\exists f.\{d\} \sqcap \neg\exists g.\{d\})^{\mathcal{I}} = \emptyset$ for any value d, and $\mathcal{I}$ cannot satisfy $x : \exists f.\{d\} \sqcap \neg\exists g.\{d\}$. For the converse, if $\mathcal{I}$ is a model of $\mathcal{E}(\mathcal{K})$ in which for some v, i, $(v,i) \in f^{\mathcal{I}}$ but $(v,i) \notin g^{\mathcal{I}}$, then $\mathcal{I}$ can be extended to $\mathcal{I}'$ such that $x^{\mathcal{I}'} = v$ and $d^{\mathcal{I}'} = i$, so $x^{\mathcal{I}'} \in (\exists f.\{d\})^{\mathcal{I}'}$, $x^{\mathcal{I}'} \in (\neg\exists g.\{d\})^{\mathcal{I}'}$ and $x^{\mathcal{I}'} \in (\exists f.\{d\} \sqcap \neg\exists g.\{d\})^{\mathcal{I}'}$. $\mathcal{I}'$ therefore satisfies $x : \exists f.\{d\} \sqcap \neg\exists g.\{d\}$, and it is still a model of $\mathcal{E}(\mathcal{K})$ because neither x nor d is mentioned in $\mathcal{E}(\mathcal{K})$, so $\mathcal{I}'$ is a model of $\mathcal{E}(\mathcal{K}) \cup \{x : \exists f.\{d\} \sqcap \neg\exists g.\{d\}\}$.*

– *$\mathcal{E}(\mathcal{K}) \models a = b$ iff $\mathcal{E}(\mathcal{K}) \cup \{a \neq b\}$ is not satisfiable. If $\mathcal{E}(\mathcal{K}) \models a = b$, then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $a^{\mathcal{I}} = b^{\mathcal{I}}$, so $\mathcal{I}$ cannot satisfy $a \neq b$. For the converse, if $\mathcal{E}(\mathcal{K}) \cup \{a \neq b\}$ is not satisfiable, then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $a^{\mathcal{I}} = b^{\mathcal{I}}$, so $\mathcal{E}(\mathcal{K}) \models a = b$.*

– *$\mathcal{E}(\mathcal{K}) \models a \neq b$ iff $\mathcal{E}(\mathcal{K}) \cup \{a = b\}$ is not satisfiable. This is a trivial variant of the previous case.* □

Theorems 1 and 2 imply:

Corollary 1. *OWL DL entailment can be transformed into knowledge base unsatisfiability in $\mathcal{SHOIN}(\mathbf{D})$.*

4.3 Consequences

The overall translation from OWL DL entailment to $\mathcal{SHOIN}(\mathbf{D})$ can be performed in polynomial time and results in a polynomial number of knowledge base satisfiability problems each of which is polynomial in the size of the initial OWL DL entailment. Therefore we have shown that OWL DL entailment is in the same complexity class as knowledge base satisfiability in $\mathcal{SHOIN}(\mathbf{D})$.

Unfortunately, $\mathcal{SHOIN}(\mathbf{D})$ is a difficult description logic. Most problems in $\mathcal{SHOIN}(\mathbf{D})$, including knowledge base satisfiability, are in NExpTime [17]. Further, there are as yet no known optimized inference algorithms or implemented systems for $\mathcal{SHOIN}(\mathbf{D})$.

The situation is not, however, completely bleak. There is an inexact translation from $\mathcal{SHOIN}(\mathbf{D})$ to $\mathcal{SHIN}(\mathbf{D})$ that turns nominals into atomic concept names. I.e., for

OWL fragment F	Translation $\mathcal{F}'(F)$
Individual($x_1 \ldots x_n$)	$\mathcal{F}'(a : x_1), \ldots, \mathcal{F}'(a : x_n)$
	for a a fresh individual name
a : type(C)	$a : \mathcal{V}(C)$
a : value($R\ x$)	$\langle a, b \rangle : R$, $\mathcal{F}'(b : x)$
	for b a fresh individual name
a : value($U\ v$)	$\langle a, v \rangle : U$
$a : o$	$a = o$

Fig. 8. Translation from OWL Lite facts to $\mathcal{SHIF}^+(\mathbf{D})$

each nominal o, occurrences of o are replaced by a new concept P_o, and an axiom $o : P_o$ is added to the knowledge base; and for each axiom $a \neq b$, the axiom $P_a \sqsubseteq \neg P_b$ is added to the knowledge base. This translation could be used to produce a partial, but still very capable, reasoner for OWL DL. Moreover, as is shown in the next section, the situation for OWL Lite is significantly different.

5 Transforming OWL Lite

OWL Lite is the subset of OWL DL that

1. eliminates the `intersectionOf`, `unionOf`, `complementOf`, and `oneOf` constructors;
2. removes the `value` construct from the `restriction` constructors;
3. limits cardinalities to 0 and 1;
4. eliminates the `enumeratedClass` axiom; and
5. requires that description-forming constructors not occur in other description-forming constructors.

The reason for defining the OWL Lite subset of OWL DL was to have an easier target for implementation. This was thought to be mostly easier parsing and other syntactic manipulations.

As OWL Lite does not have the analogue of nominals it is possible that inference is easier in OWL Lite than in OWL DL. However, the transformation above from OWL DL entailment into $\mathcal{SHOIN}(\mathbf{D})$ unsatisfiability uses nominals even for OWL Lite constructs. It is thus worthwhile to devise an alternative translation that avoids nominals.

There are three places that nominals show up in our transformation:

1. translations into $\mathcal{SHOIN}^+(\mathbf{D})$ of OWL DL constructs that are not in OWL Lite, in particular the `oneOf` constructor;
2. translations into $\mathcal{SHOIN}^+(\mathbf{D})$ axioms of OWL DL `Individual` facts; and
3. the transformation to $\mathcal{SHOIN}(\mathbf{D})$ unsatisfiability of $\mathcal{SHOIN}^+(\mathbf{D})$ entailments whose consequents are role inclusion axioms or role transitivity axioms.

The first of these, of course, is not a concern when considering OWL Lite.

Axiom A	Transformation $\mathcal{G}(A)$
$a : C$	$a : \neg C$
$\langle a, b\rangle : R$	$b : B$, $a : \forall R.\neg B$ for B a fresh concept name
$\langle a, v\rangle : U$	$a : \forall U.\overline{v}$

Fig. 9. Extended Transformation from Entailment to Unsatisfiability

The second place where nominals show up is in the translation of OWL `Individual` facts into $\mathcal{SHOIN}(\mathbf{D})$ axioms (Figure 6). In order to avoid introducing nominals, we can use the alternative translation $\mathcal{F}'$ from OWL Lite facts to $\mathcal{SHIF}^+(\mathbf{D})$ given in Figure 8. Note that, in this case, the translation $\mathcal{V}(C)$ does not introduce any nominals as we are translating OWL Lite classes.

The new transformation does, however, introduce axioms of the form $a : C$, $\langle a, b\rangle : R$ and $\langle a, v\rangle : U$ that we will need to deal with when transforming from entailment to satisfiability. We can do this by extending the transformation $\mathcal{G}$ given in Figure 7 as shown in Figure 9. The extension deals with axioms of the form $\langle a, v\rangle : U$ using a datatype derived from the negation of a data value (written $\overline{v}$), and with axioms of the form $\langle a, b\rangle : R$ using a simple transformation, described in more detail by Horrocks *et al.* [12]. This transformation exploits the fact that a fresh concept name (i.e., a concept name that is not already mentioned in the knowledge base) can be used to simulate a nominal in some cases. In particular, if B is a fresh concept name, and we assert that a is an instance of B, then any model $\mathcal{I}$ of a knowledge base $\mathcal{K}$ can be extended to a model $\mathcal{I}'$ of $\mathcal{K}$ in which $B^{\mathcal{I}'} = \{a^{\mathcal{I}'}\}$, i.e., a model in which $B \equiv \{a\}$. When using this technique, concepts such as B are called *pseudo nominals*.

The third and final place where nominals show up is in the transformation of entailments whose consequents are object role inclusion axioms or role transitivity axioms. Both these cases can also be dealt with using pseudo nominals. Object role inclusion axioms can be dealt with using a pseudo nominal transformation similar to those given in Figure 9. In this transformation, an axiom of the form $r \sqsubseteq s$ is transformed into the axiom $x : B \sqcap \exists r(\forall s^-.\neg B)$, where B is is a fresh concept name. Similarly, transitivity axioms can be dealt with by transforming an axiom $\mathsf{Trans}(r)$ into an axiom $x : B \sqcap \exists r(\exists r(\forall r^-.\neg B))$.

We will use $\mathcal{G}'$ to denote the transformation described in Figures 7 and 9 with role inclusion and transitivity transformations modified as described above.

Theorem 3. *The translation from OWL Lite to $\mathcal{SHIF}^+(\mathbf{D})$ preserves satisfiability. That is, an OWL Lite axiom or fact is satisfied by an interpretation $\mathcal{I}$ if and only if the translation is satisfied by $\mathcal{I}$.*[8]

Proof. A simple recursive argument based on the semantics of OWL Lite and $\mathcal{SHIF}^+(\mathbf{D})$ shows that the extension of OWL Lite classes, data ranges, and pieces of Individual facts is maintained in the translation. Similarly, a simple semantics based

[8] This again ignores the minor differences between OWL Lite interpretations and $\mathcal{SHIF}^+(\mathbf{D})$ interpretations.

argument shows that the translation of OWL Lite axioms and facts preserves satisfaction.

Theorem 4. *Let $\mathcal{K}$ and $\mathcal{K}'$ be $\mathcal{SHIF}^+(\mathbf{D})$ knowledge bases derived from OWL Lite ontologies. Then $\mathcal{K} \models \mathcal{K}'$ iff the $\mathcal{SHIF}(\mathbf{D})$ knowledge base $\mathcal{E}(\mathcal{K}) \cup \{\mathcal{G}'(\mathcal{A})\}$ is unsatisfiable for every axiom A in $\mathcal{K}'$.*

Proof. *As a $\mathcal{SHIF}^+(\mathbf{D})$ knowledge base is obviously a $\mathcal{SHOIN}^+(\mathbf{D})$ knowledge base, we only need to consider the new transformations introduced in $\mathcal{G}'$.*

- *$\mathcal{E}(\mathcal{K}) \models a : C$ iff $\mathcal{E}(\mathcal{K}) \cup \{a : \neg C\}$ is not satisfiable. If $\mathcal{E}(\mathcal{K}) \models a : C$ then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\mathcal{I}$ cannot satisfy $a : \neg C$. For the converse, if $\mathcal{I}$ is a model of $\mathcal{E}(\mathcal{K})$ in which $a^{\mathcal{I}} \notin C^{\mathcal{I}}$, then $\mathcal{I}$ also satisfies $a : \neg C$ and $\mathcal{E}(\mathcal{K}) \cup \{a : \neg C\}$ is satisfiable.*
- *$\mathcal{E}(\mathcal{K}) \models \langle a, b\rangle : R$ iff $\mathcal{E}(\mathcal{K}) \cup \{b : B,\ a : \forall R.\neg B\}$ is not satisfiable, where B is a concept name not mentioned in $\mathcal{E}(\mathcal{K})$. If $\mathcal{E}(\mathcal{K}) \models \langle a, b\rangle : R$, then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$, and if $b : B$ is satisfied then $a : \forall R.\neg B$ is not satisfied. For the converse, if $\mathcal{I}$ is a model of $\mathcal{E}(\mathcal{K})$ in which $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin R^{\mathcal{I}}$, then $\mathcal{I}$ can be extended to $\mathcal{I}'$ such that $B^{\mathcal{I}'} = \{b^{\mathcal{I}}\}$, so for any $(a^{\mathcal{I}'}, w) \in R^{\mathcal{I}'}$, $w \in (\neg B)^{\mathcal{I}'}$, and thus $a^{\mathcal{I}'} \in (\forall R.\neg B)^{\mathcal{I}'}$. $\mathcal{I}'$ therefore satisfies both $b : B$ and $a : \forall R.\neg B$, and it is still a model of $\mathcal{E}(\mathcal{K})$ because B is not mentioned in $\mathcal{E}(\mathcal{K})$, so $\mathcal{I}'$ is a model of $\mathcal{E}(\mathcal{K}) \cup \{b : B,\ a : \forall R.\neg B\}$.*
- *$\mathcal{E}(\mathcal{K}) \models \langle a, v\rangle : U$ iff $\mathcal{E}(\mathcal{K}) \cup \{a : \forall U.\overline{v}\}$ is not satisfiable, where $\overline{v}$ is a datatype such that $(\overline{v})^{\mathbf{D}} = \Delta^{\mathcal{I}}_{\mathbf{D}} \setminus \{v^{\mathbf{D}}\}$. If $\mathcal{E}(\mathcal{K}) \models \langle a, v\rangle : U$, then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $(a^{\mathcal{I}}, v^{\mathbf{D}}) \in U^{\mathcal{I}}$ and $a : \forall U.\overline{v}$ is not satisfied. For the converse, if $\mathcal{I}$ is a model of $\mathcal{E}(\mathcal{K})$ in which $(a^{\mathcal{I}}, v^{\mathbf{D}}) \notin U^{\mathcal{I}}$, then for any $(a^{\mathcal{I}}, i) \in U^{\mathcal{I}}$, $i \in (\overline{v})^{\mathbf{D}}$ and $a^{\mathcal{I}} \in (\forall U.\overline{v})^{\mathcal{I}}$. $\mathcal{I}$ therefore satisfies $a : \forall U.\overline{v}$, and so it is a model of $\mathcal{E}(\mathcal{K}) \cup \{a : \forall U.\overline{v}\}$.*
- *$\mathcal{E}(\mathcal{K}) \models r \sqsubseteq s$ iff $\mathcal{E}(\mathcal{K}) \cup \{x : B \sqcap \exists r(\forall s^-.\neg B)\}$ is not satisfiable, where B is a concept name not mentioned in $\mathcal{E}(\mathcal{K})$. If $\mathcal{E}(\mathcal{K}) \models r \sqsubseteq s$, then $(v, w) \in r^{\mathcal{I}}$ implies $(v, w) \in s^{\mathcal{I}}$ and $(w, v) \in (s^-)^{\mathcal{I}}$, so $x^{\mathcal{I}} \in (\exists r(\forall s^-.\neg B))^{\mathcal{I}}$ implies $x^{\mathcal{I}} \in (\neg B)^{\mathcal{I}}$ and $x : B \sqcap \exists r(\forall s^-.\neg B)$ is not satisfied. For the converse, if $\mathcal{I}$ is a model of $\mathcal{E}(\mathcal{K})$ in which for some v, w, $(v, w) \in r^{\mathcal{I}}$ but $(v, w) \notin s^{\mathcal{I}}$ (and so $(w, v) \notin (s^-)^{\mathcal{I}}$), then $\mathcal{I}$ can be extended to $\mathcal{I}'$ such that $x^{\mathcal{I}'} = v$ and $B^{\mathcal{I}'} = \{v\}$, so $x^{\mathcal{I}'} \in B^{\mathcal{I}'}$, $w \in (\forall s^-.\neg B)^{\mathcal{I}'}$ and $x^{\mathcal{I}'} \in (\exists r(\forall s^-.\neg B))^{\mathcal{I}'}$. $\mathcal{I}'$ therefore satisfies $x : B \sqcap \exists r(\forall s^-.\neg B)$, and it is still a model of $\mathcal{E}(\mathcal{K})$ because neither x nor B is mentioned in $\mathcal{E}(\mathcal{K})$, so $\mathcal{I}'$ is a model of $\mathcal{E}(\mathcal{K}) \cup \{x : B \sqcap \exists r(\forall s^-.\neg B)\}$.*
- *$\mathcal{E}(\mathcal{K}) \models \mathsf{Trans}(r)$ iff $\mathcal{E}(\mathcal{K}) \cup \{x : B \sqcap \exists r(\exists r(\forall r^-.\neg B))\}$ is not satisfiable, where B is a concept name not mentioned in $\mathcal{E}(\mathcal{K})$. If $\mathcal{E}(\mathcal{K}) \models \mathsf{Trans}(r)$, then in every model $\mathcal{I}$ of $\mathcal{E}(\mathcal{K})$, $\{(x^{\mathcal{I}}, w), (w, z)\} \subseteq r^{\mathcal{I}}$ implies $(x^{\mathcal{I}}, z) \in r^{\mathcal{I}}$ and $(z, x^{\mathcal{I}}) \in (r^-)^{\mathcal{I}}$, so $x^{\mathcal{I}} \in (\exists r(\exists r(\forall r^-.\neg B)))^{\mathcal{I}}$ implies $x^{\mathcal{I}} \in (\neg B)^{\mathcal{I}}$ and $x : B \sqcap \exists r(\exists r(\forall r^-.\neg B))$ is not satisfied. For the converse, if $\mathcal{I}$ is a model of $\mathcal{E}(\mathcal{K})$ in which for some v, w, z, $\{(v, w), (w, z)\} \subseteq r^{\mathcal{I}}$ but $(v, z) \notin r^{\mathcal{I}}$ (and so $(z, v) \notin (r^-)^{\mathcal{I}}$), then $\mathcal{I}$ can be extended to $\mathcal{I}'$ such that $x^{\mathcal{I}'} = v$ and $B^{\mathcal{I}'} = \{v\}$, so $x^{\mathcal{I}'} \in B^{\mathcal{I}'}$, $z \in (\forall r^-.\neg B)^{\mathcal{I}'}$ and $x^{\mathcal{I}'} \in (\exists r(\exists r(\forall r^-.\neg B)))^{\mathcal{I}'}$. $\mathcal{I}'$ therefore satisfies $x : B \sqcap \exists r(\exists r(\forall r^-.\neg B))$, and it is still a model of $\mathcal{E}(\mathcal{K})$ because neither x nor B is mentioned in $\mathcal{E}(\mathcal{K})$, so $\mathcal{I}'$ is a model of $\mathcal{E}(\mathcal{K}) \cup \{x : B \sqcap \exists r(\exists r(\forall r^-.\neg B))\}$.* □

Theorems 3 and 4 imply:

Corollary 2. *OWL Lite entailment can be transformed into knowledge base unsatisfiability in* $\mathcal{SHIF}(\mathbf{D})$.

A simple examination shows that the transformations can be computed in polynomial time and result in only a linear increase in size.

As knowledge base satisfiability in $\mathcal{SHIF}(\mathbf{D})$ is in EXPTIME [17], this means that entailment in OWL Lite can be computed in exponential time. Further, OWL Lite entailment can be computed by the RACER description logic system [8], a heavily-optimised description logic reasoner, resulting in an effective reasoner for OWL Lite entailment.

6 Conclusion

Reasoning with ontology languages will be important in the Semantic Web if applications are to exploit the semantics of ontology based metadata annotations.

We have shown that ontology entailment in the OWL DL and OWL Lite ontology languages can be reduced to knowledge base satisfiability in, respectively, the $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$ description logics. This is so even though some constructs in these languages go beyond the standard description logic constructs.

From these mappings, we have determined that the complexity of ontology entailment in OWL DL and OWL Lite is in NEXPTIME and EXPTIME respectively (the same as for knowledge base satisfiability in $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$ respectively). The mapping of OWL Lite to $\mathcal{SHIF}(\mathbf{D})$ also means that already-known practical reasoning algorithms for $\mathcal{SHIF}(\mathbf{D})$ can be used to determine ontology entailment in OWL Lite; in particular, the highly optimised RACER system [8], which can determine knowledge base satisfaction in $\mathcal{SHIF}(\mathbf{D})$, can be used to provide efficient reasoning services for OWL Lite.

The mapping from OWL DL to $\mathcal{SHOIN}(\mathbf{D})$ can also be used to provide complete reasoning services for a large part of OWL DL, or partial reasoning services for all of OWL DL. Studies directed towards the development of complete, practical algorithms and systems for all of OWL DL are obviously a high priority within the description logic and Semantic Web research communities. If such algorithms cannot be found, it may be worthwhile to consider revising the specification of OWL DL to eliminate (or at least weaken) one of the constructors whose interaction causes the difficulty, i.e., inverse properties, cardinality constraints or oneOf.

Bibliography

[1] Dave Beckett. RDF/XML syntax specification (revised). W3C Recommendation, 2004. Available at `http://www.w3.org/TR/rdf-syntax-grammar/`.

[2] Tim Berners-Lee. *Weaving the Web*. Harpur, San Francisco, 1999.

[3] Paul V. Biron and Ashok Malhotra. XML schema part 2: Datatypes. W3C Recommendation, 2001. Available at `http://www.w3.org/TR/2003/WD-xmlschema-2-20010502/`.

[4] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language reference. W3C Recommendation, 2004. Available at `http://www.w3.org/TR/owl-ref/`.

[5] S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. Klein, and J. Broekstra. Knowledge representation on the web. In *Proc. of the 2000 Description Logic Workshop (DL 2000)*, pages 89–98, 2000.

[6] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.

[7] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[8] Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 701–705. Springer, 2001.

[9] Patrick Hayes. RDF semantics. W3C Recommendation, 2004. Available at `http://www.w3.org/TR/rdf-mt/`.

[10] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$ description logic. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, 2001.

[11] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer, 1999.

[12] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a description logic. In *Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning (LPAR'2000)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000.

[13] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. Reviewing the design of DAML+OIL: An ontology language for the semantic web. In *Proc. of the 18th Nat. Conf. on Artificial Intelligence (AAAI 2002)*, 2002.

[14] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation, 2004. Available at `http://www.w3.org/TR/rdf-concepts/`.

[15] Ora Lassila and Ralph R. Swick. Resource description framework (RDF) model and syntax specification. W3C Recommendation, 1999. Available at `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222`.

[16] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. W3C Recommendation, 2004. Available at `http://www.w3.org/TR/owl-semantics/`.

[17] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.