Reducing OWL Entailment to Description Logic Satisfiability

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

[1] Department of Computer Science
University of Manchester
horrocks@cs.man.ac.uk
[2] Bell Labs Research
Lucent Technologies
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 [10] and OWL [4]. OWL is of particular significance as it has been developed by the W3C Web Ontology working group, and is set to become a W3C recommendation.

The proposed 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 are basically very expressive description logics with an RDF syntax. They can therefore exploit the considerable existing body of description logic research, e.g., to define the semantics of the language and to understand its formal properties, in particular the decidability and complexity of key inference 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

D. Fensel et al. (Eds.): ISWC 2003, LNCS 2870, pp. 17–29, 2003.

much harder to compute.[1] In this paper we will, therefore, concentrate on the provision of reasoning services for OWL Lite and OWL DL.

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, e.g., if semantic search engines are to find pages based on the semantics of their annotations rather than their syntax. 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.

The obvious solution to the first problem 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 problem 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 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 (resp. 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. Unfortunately, the design of "practical" algorithms for $\mathcal{SHOIN}(\mathbf{D})$ is still an open problem – the search for such algorithms must obviously be a high priority within the Semantic Web research community.

[1] 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 [13].

2 The OWL Web Ontology Language

As mentioned in Section 1, OWL [4] 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.

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 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.

The constructors used to form OWL DL descriptions and data ranges are summarised in Figure 1, where A is a class name, C (possibly subscripted) is a description, o (possibly subscripted) is an individual name, R is an object (or abstract) property, T is a datatype property,[2] B is a datatype, D is a data range, v (possibly subscripted) is a data value and ℓ, m, n are non-negative integers; elements {enclosed in braces} can be repeated zero or more times and elements [enclosed in square brackets] are optional. The details of these constructors can be found in the OWL documentation [4].

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

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(ℓ)])
restriction(T
{allValuesFrom(D)} {someValuesFrom(D)}
{value(v)} [minCardinality(n)]
[maxCardinality(m)] [cardinality(ℓ)])
Data Ranges
B
oneOf($v_1 \ldots v_n$)

Fig. 1. OWL DL Constructors

Descriptions and data ranges can be used in OWL DL axioms and facts to provide information about classes, properties, and individuals. Figure 2 provides a summary of these axioms and facts. The details of these constructors can also be found in the OWL documentation [4]. In particular, 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 semantic 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]. Again, we will not present all of this semantics, instead concentrating on its differences from the usual description logics semantics.

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` constuctor (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 [11]. In Section 5 we will examine these differences in more detail, and explore their impact on the reduction from OWL entailment to description logic satisfiability.

2.2 Semantics for OWL DL

The semantics for OWL DL is fairly standard by description logic standards. The OWL semantic domain is a set whose elements can be divided into abstract objects (the abstract domain), and datatype values (the datatype or concrete domain, written $\Delta^{\mathcal{I}}_{\mathbf{D}}$). Datatypes

Class Axioms
Class(A partial $d_1 \ldots d_n$)
Class(A complete $d_1 \ldots d_n$)
EnumeratedClass(A $o_1 \ldots o_n$)
DisjointClasses($d_1 \ldots d_n$)
EquivalentClasses($d_1 \ldots d_n$)
SubClassOf(d_1 d_2)
Property Axioms
DatatypeProperty(U super(U_1) ... super(U_n) [Functional] domain(d_1) ... domain(d_m) range(r_1) ... domain(r_l))
ObjectProperty(P super(P_1) ... super(P_n) [inverseOf(P_0)] [Functional] [InverseFunctional] [Symmetric] [Transitive] domain(d_1) ... domain(d_m) range(e_1) ... domain(e_l))
EquivalentProperties($U_1 \ldots U_n$)
SubPropertyOf(U_1 U_2)
EquivalentProperties($P_1 \ldots P_n$)
SubPropertyOf(P_1 P_2)
Facts
Individual([o] type(d_1) ... type(d_1) value(p_1 v_1) ... value(p_1 v_1))
SameIndividual($o_1 \ldots o_n$)
DifferentIndividuals($o_1 \ldots o_n$)

Fig. 2. OWL DL Axioms and Facts (simplified)

in OWL are derived from the built-in XML Schema datatypes [3]. Datatype values are denoted by special literal constructs in the syntax, the details of which need not concern us here.

An interpretation in this semantics is officially a four-tuple consisting of the abstract domain and separate mappings for concept names, property names, and individual names (in description logics, the mappings are usually combined to give a two-tuple, but the two forms are obviously equivalent). OWL DL classes are interpreted as subsets of the abstract domain, and for each constructor the semantics of the resulting class is defined in terms of its components. For example, given two classes C and D, the interpretation of the intersection of C and D is defined to be the intersection of the interpretations of C and D. Datatypes are handled by means of a mapping $.^{\mathbf{D}}$ that interprets datatype names as subsets of the concrete domain and data names (i.e., lexical representations of data values) as elements of the concrete domain.

OWL DL axioms and facts result in semantic conditions on interpretations. For example, an axiom asserting that C is a subclass of D results in the semantic condition that the interpretation of C must be a subset of the interpretation of D, while a fact asserting that a has type C results in the semantic condition that the interpretation of a must be an element of the set that is the interpretation of C. 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 $\mathcal{I}$.

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 $O_1 \models O_2$,

Constructor Name	Syntax	Semantics
atomic concept $\mathbf{A}$	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
datatypes $\mathbf{D}$	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 $\mathbf{I}$	o	$o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
data values	v	$v^{\mathcal{I}} = v^{\mathbf{D}}$
inverse role	R^-	$(R^-)^{\mathcal{I}} = (R^{\mathcal{I}})^-$
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	$\neg C_1$	$(\neg C_1)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C_1^{\mathcal{I}}$
oneOf	$\{o_1, \ldots\}$	$\{o_1, \ldots\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, \ldots\}$
exists restriction	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y. \langle x \rangle y \in R^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$
value restriction	$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y. \langle x, y \rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
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	$\exists U.D$	$(\exists U.D)^{\mathcal{I}} = \{x \mid \exists y. \langle x, y \rangle \in U^{\mathcal{I}}$ and $y \in D^{\mathbf{D}}\}$
datatype value	$\forall U.D$	$(\forall U.D)^{\mathcal{I}} = \{x \mid \forall y. \langle x, y \rangle \in U^{\mathcal{I}} \rightarrow y \in D^{\mathbf{D}}\}$
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 $\mathcal{SHOIN}^+(\mathbf{D})$

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.

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

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 [11], but is extended with inverse roles ($\mathcal{I}$) and restricted to unqualified number restrictions ($\mathcal{N}$). We will assume throughout the paper that datatypes and data values are as in OWL.

Let $\mathbf{A}, \mathbf{R}_A, \mathbf{R_D}$, and $\mathbf{I}$ be pairwise disjoint sets of *concept names*, *abstract role names*, *datatype (or concrete) role names*, and *individual names*. The set of $\mathcal{SHOIN}(\mathbf{D})$-roles is $\mathbf{R}_A \cup \{R^- \mid R \in \mathbf{R}_A\} \cup \mathbf{R_D}$. 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.

The $\mathcal{SHOIN}(\mathbf{D})$ axiom syntax is also given in Figure 3. (The last axiom in Figure 3 forms an extension of $\mathcal{SHOIN}(\mathbf{D})$, which we call $\mathcal{SHOIN}^+(\mathbf{D})$, which is used internally in our translation.) A *knowledge base* $\mathcal{K}$ is a finite set of axioms. We will use $\sqsubseteq^*$ to denote the transitive reflexive closure of $\sqsubseteq$ on roles, i.e., for two roles S, R in $\mathcal{K}$, $S \sqsubseteq^* R$ 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 [13].

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 (or concrete) 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 $\mathcal{K}$; $\mathcal{K}$ is *satisfiable* (*unsatisfiable*) iff there exists (does not exist) such an interpretation. A $\mathcal{SHOIN}(\mathbf{D})$-concept C is satisfiable w.r.t. a knowledge base $\mathcal{K}$ iff there is an interpretation $\mathcal{I}$ with $C^\mathcal{I} \neq \emptyset$ that satisfies $\mathcal{K}$. A concept C is *subsumed* by a concept D w.r.t. $\mathcal{K}$ iff $C^\mathcal{I} \sqsubseteq D^\mathcal{I}$ for each interpretation $\mathcal{I}$ satisfying $\mathcal{K}$. Two concepts are said to be equivalent w.r.t. $\mathcal{K}$ iff they subsume each other w.r.t. $\mathcal{K}$. A knowledge base $\mathcal{K}_1$ entails a knowledge base $\mathcal{K}_2$ iff every interpretation of $\mathcal{K}_1$ is also an interpretation of $\mathcal{K}_2$.

We define a notion of entailment in $\mathcal{SHOIN}^+(\mathbf{D})$ in the same way as it was defined for OWL DL. 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. $\mathcal{SHIF}^+(\mathbf{D})$ is related to $\mathcal{SHOIN}^+(\mathbf{D})$ in the same way.

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.)

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(U v)	$\exists U.\{v\}$
o	$\{o\}$

Fig. 4. Translation from OWL facts 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(`$C_1 \ldots C_n$`)` 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 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 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 4 describes a translation $\mathcal{F}$ that transforms OWL facts into a $\mathcal{SHOIN}^+(\mathbf{D})$ existence axioms, where C is an OWL class, R is an OWL abstract property or $\mathcal{SHOIN}^+(\mathbf{D})$ abstract role, U is an OWL datatype property or $\mathcal{SHOIN}^+(\mathbf{D})$ datatype role, o is an individual name, v is a data value, and $\mathcal{V}$ is the above mentioned translation from OWL classes to $\mathcal{SHOIN}^+(\mathbf{D})$ concepts.

Theorem 1. *The translation from OWL DL to $\mathcal{SHOIN}^+(\mathbf{D})$ preserves equivalence. That is, an OWL DL axiom or fact is satisfied by an interpretation $\mathcal{I}$ if and only if the translation is satisfied by $\mathcal{I}$.*[3]

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.

[3] 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.

Axiom A	Transformation $\mathcal{G}(A)$
$c \sqsubseteq d$	$x : c \sqcap \neg d$
$\exists c$	$\top \sqsubseteq \neg c$
$\mathsf{Trans}(r)$	$x : \exists r.\exists r.\{y\} \sqcap \neg \exists r.\{y\}$
$r \sqsubseteq s$	$x : \exists r.\{y\} \sqcap \neg \exists s.\{y\}$
$f \sqsubseteq g$	$x : \bigsqcup_{z \in \mathbf{V}} \exists f.\{z\} \sqcap \neg \exists g.\{z\}$ for $\mathbf{V}$ = the set of data values in $\mathcal{K}$, plus one fresh data value for each datatype in $\mathcal{K}$
$a = b$	$a \neq b$
$a \neq b$	$a = b$

Fig. 5. Translation from Entailment to Unsatisfiability

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 5) 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 that requires the existence of an individual that is related to some other individual by r but not by s, thus violating the axiom. The only unusual translation is for datatype role inclusions $f \sqsubseteq g$. Because data values have a known "identity" (rather like individuals under the unique name assumption), a fresh value cannot be used to simulate an existentially quantified variable that could be interpreted as any element in the datatype domain (in the way the fresh nominal is used in the case of an object role inclusion axiom). Instead, it is necessary to show that the relevant inclusion holds for every data value that occurs in the knowledge base, plus one fresh data value (i.e., one that does not occur in the knowledge base) for each datatype in $\mathcal{K}$. Because there are no operations on data values, it suffices to consider only these fresh data values in addition to those that occur in the knowledge base.

The translation $\mathcal{G}$ increases the size of an axiom to at most the larger of its size and the size of the knowledge base. It can easily be performed in time linear in the larger of the size of the axiom and the size of the knowledge base. (If datatype role inclusions are not used, then $\mathcal{G}$ increases the size of an axiom by atmost a constant amount.)

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 existance 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

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

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 $\mathcal{K}'$.*

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. 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.

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. 7. Extended Transformation from Entailment to Unsatisfiability

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.

The second place where nominals show up is in the translation of OWL `Individual` facts into $\mathcal{SHOIN}(\mathbf{D})$ axioms (Figure 4). In order to avoid introducing nominals, we can use the alternative transformation $\mathcal{F}'$ given in Figure 6. 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 5 as shown in Figure 7. The extension deals with axioms of the form $\langle a, b\rangle : R$ using a simple transformation, described in more detail by [12], and with axioms of the form $\langle a, v\rangle : U$ using a datatype derived from the negation of a data value (written $\overline{v}$).

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.

Object role inclusion axioms can be dealt with using a transformation similar to those given in Figure 7 (and described in more detail in [12]), which does not introduce any nominals. This is shown in the following lemma:

Lemma 1. *Let K be an OWL Lite ontology and let A be an OWL Lite role inclusion axiom stating that r is a subrole of s. Then $K \models A$ iff $\mathcal{E}(K) \cup \{x : B \sqcap \exists r(\forall s^-.\neg B)\}$ is unsatisfiable for x a fresh individual name, and B a fresh concept name.*

Transitivity axioms can be dealt with by exploiting the more limited expressive power of OWL Lite, in particular its inability to describe classes, datatypes or properties whose interpretations must be non-empty but finite (e.g., classes described using the `oneOf` constructor). As a result of this more limited expressive power, the only way to deduce the transitivity of a property r is to show that the interpretation of r cannot form any chains (i.e., consists only of isolated tuples, or is empty). This observation leads to the following lemma:

Lemma 2. *Let K be an OWL Lite ontology and let A be an OWL Lite role transitivity axiom stating that r is transitive. Then* $K \models A$ *iff* $\mathcal{E}(K) \cup \{x : \exists r(\exists r \top)\}$ *is unsatisfiable for* x *a fresh individual name (i.e.,* r *forms no chains).*

The above lemmas, taken together, show that OWL Lite entailment can be transformed into knowledge base unsatisfiability in $\mathcal{SHIF}(\mathbf{D})$, plus some simple (and easy) tests on the syntactic form of a knowledge base. 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. In spite of its known decidability, however, the design of "practical" decision procedures for $\mathcal{SHOIN}(\mathbf{D})$ is still an open problem. The search for such algorithms must obviously be a high priority within the Semantic Web research community.

References

[1] D. Beckett. RDF/XML syntax specification (revised). W3C Working Draft, 2003. Available at `http://www.w3.org/TR/2003/WD-rdf-syntax-grammar-20030123`.

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

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

[4] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Web ontology language (OWL) reference version 1.0. W3C Working Draft, 2003. Available at `http://www.w3.org/TR/2003/WD-owl-ref-20030331`.

[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] F. M. Donini, M. Lenzerini, D. Nardi, and W. 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] V. Haarslev and R. 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] P. Hayes. RDF semantics. W3C Working Draft, 2003. Available at `http://www.w3.org/TR/2003/WD-rdf-mt-20030123`.

[10] I. Horrocks, P. F. Patel-Schneider, and F. 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.

[11] 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.

[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] I. Horrocks, U. Sattler, and S. 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.

[14] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C Working Draft, 2003. Available at `http://www.w3.org/TR/2003/WD-rdf-concepts-20030123`.

[15] O. Lassila and R. 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] P. F. Patel-Schneider, P. Hayes, I. Horrocks, and F. van Harmelen. Web ontology language (OWL) abstract syntax and semantics. W3C Working Draft, 2003. Available at `http://www.w3.org/TR/2003/WD-owl-ref-20030331`.

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