Reducing OWL Entailment to Description Logic Satisfiability

Ian Horrocks
Department of Computer Science
University of Manchester, Manchester, UK

Peter F. Patel-Schneider
Bell Labs Research, Lucent Technologies
Murray Hill, NJ, USA

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.

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. 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 [8]. The importance of ontologies in semantic markup has prompted the development of several ontology languages specifically designed for this purpose, leading up to OWL [2], currently being developed by the W3C Web Ontology working group. Reasoning with ontology languages will be important in the Semantic Web if applications are to exploit the semantics of ontology based metadata annotations. It is thus useful to make connections between OWL and logics that have effective reasoners.

In this paper we will show that the main reasoning problem (ontology entailment) in OWL DL and OWL Lite (two subsets of OWL) can be reduced to knowledge base (KB) satisfiability in the $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$ description logics respectively. 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 [4]) 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.)

2 The OWL Web Ontology Language

OWL is defined as an extension to RDF in the form of a vocabulary entailment [5], i.e., the syntax of OWL is the syntax of RDF and the semantics of OWL are an extension of the

Classes (C)
A (class name)
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 (D)
B (datatype name)
oneOf($v_1 \ldots v_n$)

Figure 1: OWL DL Constructors

semantics of RDF. OWL has many features in common with description logics, but also has two significant differences. The first difference between OWL and description logics is that OWL information is encoded in RDF/XML documents [1] and parsed into RDF Graphs [7] 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. 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.

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 to the point that it is possible to develop an abstract syntax for OWL DL [9] that is not very different from description logic syntaxes. 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 KBs, which are the highest level of OWL DL syntax. This syntax for OWL DL is summarised in Figure 1.

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 [2]. 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 [9].

There is a subset of OWL DL, called OWL Lite, that has fewer constructors than OWL

Class Axioms
Class(A partial $C_1 \dots C_n$)
Class(A complete $C_1 \dots C_n$)
EnumeratedClass(A $o_1 \dots o_n$)
DisjointClasses($C_1 \dots C_n$)
EquivalentClasses($C_1 \dots C_n$)
SubClassOf(C_1 C_2)
Property Axioms
DatatypeProperty(U super(U_1) ... super(U_n) [Functional] domain(C_1) ... domain(C_m) range(D_1) ... domain(D_l))
ObjectProperty(P super(P_1) ... super(P_n) [inverseOf(P_0)] [Functional] [InverseFunctional] [Symmetric] [Transitive] domain(C_1) ... domain(C_m) range(e_1) ... domain(e_l))
EquivalentProperties($U_1 \dots U_n$)
SubPropertyOf(U_1 U_2)
EquivalentProperties($P_1 \dots P_n$)
SubPropertyOf(P_1 P_2)
Facts
Individual([o] type(C_1) ... type(C_n) value(p_1 v_1) ... value(p_1 v_1))
SameIndividual($o_1 \dots o_n$)
DifferentIndividuals($o_1 \dots o_n$)

Figure 2: OWL DL Axioms and Facts (simplified)

DL and limits the use of some of these constructors. In particular, OWL Lite does not support the `oneOf` constuctor (equivalent to description logic *nominals*).

The semantics for OWL DL is fairly standard by description logic standards, with interpretations modelling axioms and ontologies. The main semantic relationship in OWL DL is entailment—a relationship between pairs of OWL ontologies.[1] 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 KB and concept satisfiability. The main goal of this paper is to show how OWL DL entailment can be transformed into DL KB (un)satisfiability.

3 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 $\mathcal{SHOIN}^+(\mathbf{D})$ ($\mathcal{SHOIN}(\mathbf{D})$ augmented with a concept non-emptiness construct, $\exists C$). Then $\mathcal{SHOIN}^+(\mathbf{D})$ entailment is transformed into

[1] The emphasis on entailment is mainly a consequence of OWL's relationship with RDF. On the one hand, RDF entailment cannot be reduced to consistency as the language does not include negation, and consistency itself is not very interesting as the language is too weak to express an inconsistency. On the other hand, the use of RDF properties to express subclass and type relationships means that subsumption and instantiation equate to entailment of the relevant RDF triple, i.e., $C \sqsubseteq D$ w.r.t. an ontology O iff O entails the RDF triple (C `subClassOf` D), and $i \in C$ w.r.t. O iff O entails the RDF triple (i `type` C).

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	$\{o\}$

Figure 3: Translation from OWL facts to $\mathcal{SHOIN}^+(\mathbf{D})$

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

Figure 4: Translation from Entailment to Unsatisfiability

unsatisfiability of $\mathcal{SHOIN}(\mathbf{D})$ KBs. (Note that concept non-emptiness axioms are eliminated in this last step, leaving a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base.)

Translating OWL DL axioms into $\mathcal{SHOIN}^+(\mathbf{D})$ is very natural, and is almost identical to the translation of OIL described by [3]. 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 [3]. 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, because facts can be stated with respect to anonymous individuals, such as the fact Individual(type(C) value(R Individual(type(D)))). The $\mathcal{SHOIN}^+(\mathbf{D})$ non-emptiness axiom can be employed, allowing the above fact to be translated into the axiom $\exists(C \sqcap \exists R.D)$. Figure 3 describes a translation $\mathcal{F}$ that transforms OWL facts into a $\mathcal{SHOIN}^+(\mathbf{D})$ non-emptiness axioms.

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 $\mathcal{I}$.*

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 KB.

The next step of our process is to transform $\mathcal{SHOIN}^+(\mathbf{D})$ knowledge base entailment to $\mathcal{SHOIN}(\mathbf{D})$ KB unsatisfiability. We define (in Figure 4) a translation, $\mathcal{G}$, such that $\mathcal{K} \models A$ iff $\mathcal{K} \cup \{\mathcal{G}(A)\}$ is unsatisfiable, for $\mathcal{K}$ a $\mathcal{SHOIN}(\mathbf{D})$ KB and A a $\mathcal{SHOIN}(\mathbf{D})$ axiom. Throughout the translation, x and y are fresh individual names.

Most of the translations in $\mathcal{G}$ are quite standard and simple. 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 KB, plus one fresh data value (i.e., one that does not occur in the KB) 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 KB.

The translation $\mathcal{G}$ increases the size of an axiom to at most the larger of its size and the size of the KB. It can easily be performed in time linear in the larger of the size of the axiom and the size of the KB.

The translation $\mathcal{G}$ eliminates concept non-emptiness axioms from the KB $\mathcal{K}'$ 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 satisfiability, can be easily performed, and only increases the size of a KB 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 $\mathcal{SHOIN}(\mathbf{D})$ KB $\mathcal{E}(\mathcal{K}) \cup \{\mathcal{G}(A)\}$ is unsatisfiable for every axiom A in $\mathcal{K}'$.*

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 KB 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 KB satisfiability, are in NEXPTIME [10]. Futher, 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 useful, reasoner for OWL DL. Moreover, as is shown in the next section, the situation for OWL Lite is significantly different.

4 Transforming OWL Lite

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.

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)	$\langle a, v \rangle : U$
$a : o$	$a = o$

Figure 5: Translation from OWL Lite facts to $\mathcal{SHIF}^+(\mathbf{D})$

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
$\langle a, v \rangle : U$	$a : \forall U.\overline{v}$

Figure 6: 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 3). In order to avoid introducing nominals, we can use the alternative transformation $\mathcal{F}'$ given in Figure 5. 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 4 as shown in Figure 6. The extension deals with axioms of the form $\langle a, b \rangle : R$ using a simple transformation, described in more detail by [6], 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 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 6 (and described in more detail in [6]), 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 KB unsatisfiability in $\mathcal{SHIF}(\mathbf{D})$, plus some simple 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 KB satisfiability in $\mathcal{SHIF}(\mathbf{D})$ is in EXPTIME [10] 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 [4], a heavily-optimised description logic reasoner, resulting in an effective reasoner for OWL Lite entailment.

5 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 KB 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 KB 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 [4], which can determine KB 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.

References

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

[2] Mike Dean, Dan Connolly, Frank van Harmelen, James Hendler, Ian 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`.

[3] 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.

[4] 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.

[5] Patrick Hayes. RDF semantics. W3C Working Draft, 2003. Available at http://www.w3.org/TR/2003/WD-rdf-mt-20030123.

[6] 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.

[7] Graham Klyne and Jeremy 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.

[8] 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.

[9] Peter F. Patel-Schneider, Patrick Hayes, Ian Horrocks, and Frank 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.

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