

Ontology Reasoning in the $\mathcal{SHOQ}(\mathbf{D})$ Description Logic

Ian Horrocks

Department of Computer Science
University of Manchester, UK
horrocks@cs.man.ac.uk

Ulrike Sattler

LuFg Theoretical Computer Science
RWTH Aachen, Germany
sattler@informatik.rwth-aachen.de

Abstract

Ontologies are set to play a key rôle in the “Semantic Web” by providing a source of shared and precisely defined terms that can be used in descriptions of web resources. Reasoning over such descriptions will be essential if web resources are to be more accessible to automated processes. $\mathcal{SHOQ}(\mathbf{D})$ is an expressive description logic equipped with named individuals and concrete datatypes which has almost exactly the same expressive power as the latest web ontology languages (e.g., OIL and DAML). We present sound and complete reasoning services for this logic.

1 Introduction

The recent explosion of interest in the World Wide Web has also fuelled interest in ontologies.¹ This is due both to the use of ontologies in existing Web based applications and to their likely rôle in the future development of the Web [van Heijst *et al.*, 1997; McGuinness, 1998; Uschold and Grüninger, 1996]. In particular, it has been predicted that ontologies will play a pivotal rôle in the *Semantic Web*—the World Wide Web Consortium’s vision of a “second generation” Web in which Web resources will be more readily accessible to automated processes [Berners-Lee, 1999].

A key component of the Semantic Web will be the annotation of web resources with meta-data that describes their content, with ontologies providing a source of shared and precisely defined terms that can be used in such meta-data. This requirement has led to the extension of Web markup languages in order to facilitate content description and the development of Web based ontologies, e.g., XML Schema, RDF (Resource Description Framework), and RDF Schema [Decker *et al.*, 2000]. RDF Schema (RDFS) in particular is recognisable as an ontology/knowledge representation language: it talks about classes and properties (binary relations), range and domain constraints (on properties), and subclass and subproperty (subsumption) relations. However, RDFS is a very primitive language (the above is an almost

complete description of its functionality), and more expressive power would clearly be necessary/desirable in order to describe resources in sufficient detail. Moreover, such descriptions should be amenable to *automated reasoning* if they are to be used effectively by automated processes.

These considerations have led to the development of OIL [Fensel *et al.*, 2000] and DAML [Hendler and McGuinness, 2001], two ontology languages that extend RDFS with a much richer set of modelling primitives. Both languages have been designed in such a way that they can be mapped onto a very expressive description logic (DL).² This mapping provides them with a formal semantics, a clear understanding of the characteristics of various reasoning problems (e.g., subsumption/satisfiability), and the possibility of exploiting existing decision procedures. OIL, in particular, was designed so that reasoning services could be provided, via a mapping to the $\mathcal{SHIQ}$ DL, by the FaCT system [Horrocks *et al.*, 1999; Horrocks, 2000].

Unfortunately, these mappings are currently incomplete in two important respects. Firstly, any practical ontology language will need to deal with *concrete datatypes* (numbers, strings, etc.) [Baader and Hanschke, 1991]. E.g., ontologies used in e-commerce may want to classify items according to weight, and to reason that an item weighing more than 50 kilogrammes is a kind of item that requires special shipping arrangements. OIL already supports the use of integers and strings in class descriptions, and it is anticipated that DAML+OIL, a new language developed from a merging of the DAML and OIL efforts, will support (most of) the datatypes defined or definable by XML Schema. However, the $\mathcal{SHIQ}$ logic implemented in the FaCT system does not include any concrete datatypes, so there is no mechanism for reasoning with this part of the language.

Secondly, realistic ontologies typically contain references to named individuals within class descriptions. E.g., “Italians” might be described as persons who are citizens of “Italy”, where Italy is a named individual [Schaerf, 1994]. The required functionality can be partially simulated by treating such individuals as pairwise disjoint atomic classes (this is the approach taken in the existing OIL $\rightarrow$ FaCT mapping), but this can result in incorrect inferences.

In this paper we will present a new DL that overcomes

¹The word ontology has been used—some would say abused—in a wide range of contexts. In this paper it will be taken to mean a formally defined model of (part of) the domain of interest.

²In fact they can be viewed as syntactic variants of such a logic.

both of the above deficiencies by taking the logic SHQ and extending it with individuals ($\mathcal{O}$) and concrete datatypes ($\mathbf{D}$) to give $SHOQ(\mathbf{D})$. The starting point for these extensions is SHQ rather than $SHIQ$ (i.e., without inverse roles), because reasoning with inverse roles is known to be difficult and/or highly intractable when combined with either concrete datatypes or named individuals: the concept satisfiability problem is known to be NExpTime hard even for the basic DL $\mathcal{ALC}$ augmented with inverse roles and either concrete datatypes or named individuals [Lutz, 2000; Tobies, 2000]. This hardness result for concrete datatypes is not yet directly applicable to $SHOQ(\mathbf{D})$ as it depends on comparisons of concrete values (binary predicates), but the addition of such comparisons would be a natural future extension to $SHOQ(\mathbf{D})$. Moreover, the presence of nominals in any DL leads to the loss of the tree/forest model property, which becomes particularly problematical in the presence of inverse roles, number restrictions, and general axioms. As a result, to the best of our knowledge, there is no (practicable) decision procedure for $SHIQ$ with nominals or converse-DPDL with nominals, the latter being a close relative of $SHIQ$ from dynamic logics [Streit, 1982]. Finally, since individuals and concrete datatypes are much more widely used in ontologies than inverse roles [Corcho and Pérez, 2000], $SHOQ(\mathbf{D})$ is a very useful addition to our reasoning armoury.

2 Preliminaries

In this section, we will describe our choice of concrete datatypes and named individuals, and introduce the syntax and semantics of $SHOQ(\mathbf{D})$.

Concrete Datatypes Concrete datatypes are used to represent literal values such as numbers and strings. A type system typically defines a set of “primitive” datatypes, such as *string* or *integer*, and provides a mechanism for deriving new datatypes from existing ones. For example, in the XML schema type system the *nonNegativeInteger* datatype is derived from the *integer* datatype by constraining values of *nonNegativeInteger* to be greater than or equal to zero [Biron and Malhorta, 2000].

In order to represent concepts such as “persons whose age is at least 21”, we can extend our concept language with a set $\mathbf{D}$ of concrete datatypes and concepts of the form $\exists R.d$ and $\forall R.d$, where $d \in \mathbf{D}$. To be more precise, we assume that we have a set of datatypes $\mathbf{D}$, and, with each $d \in \mathbf{D}$, a set $d^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}$ is associated, where $\Delta_{\mathbf{D}}$ is the domain of all datatypes. We will assume that:

1. the domain of interpretation of all concrete datatypes $\Delta_{\mathbf{D}}$ (the *concrete domain*) is disjoint from the domain of interpretation of our concept language (the *abstract domain*), and
2. there exists a sound and complete decision procedure for the emptiness of an expression of the form $d_1^{\mathbf{D}} \cap \dots \cap d_n^{\mathbf{D}}$, where d_i is a (possibly negated) concrete datatype from $\mathbf{D}$ (where $\neg d$ is interpreted as $\Delta_{\mathbf{D}} \setminus d^{\mathbf{D}}$).

We will say that a set of datatypes is *conforming* if it satisfies the above criteria.

The disjointness of the abstract and concrete domains is motivated by both philosophical and pragmatic considerations. On the one hand, concrete datatypes are considered to be already sufficiently structured by the type system, which may include a derivation mechanism and built-in ordering relations; therefore, we do not need the DL mechanism to form new datatypes as in [Baader and Hanschke, 1991]. On the other hand, it allows us to deal with an arbitrary conforming set of datatypes without compromising the compactness of our concept language or the soundness and completeness of our decision procedure.

This scheme can be trivially extended to include boolean combinations of datatypes and number restrictions qualified with data types, but to simplify the presentation we will only consider (possibly negated) atomic datatypes and exists/value restrictions. The type system can be as complex as that defined for XML schema, or as simple as the one defined in the OIL ontology language [Fensel *et al.*, 2000], where the only primitive datatypes are integer and string, and new types are derived by adding minimum and maximum value constraints. Using the OIL typesystem we could, for example, define the type (min 21) and use it in the concept $\text{Person} \sqcap \text{age}(\text{min } 21)$.

Named Individuals Allowing named individuals to occur in concepts provides additional expressive power that is useful in many applications; *nominals* (as such individuals can be called) are a prominent feature of hybrid logics [Blackburn and Seligman, 1998], and various extensions of modal and description logics with nominals have already been investigated (see, e.g., [Schaerf, 1994; De Giacomo, 1995; Areces *et al.*, 2000]). As we have seen, nominals occur naturally in ontologies as names for specific persons, companies, countries etcetera.

From a semantic point of view, it is important to distinguish between a nominal and an atomic concept/simple class, since the nominal stands for exactly one individual—in contrast to a concept, which is interpreted as some *set* of individuals. Modelling nominals as pairwise disjoint atomic concepts can lead to incorrect inferences, in particular with respect to implicit maximum cardinality constraints. For example, if Italy is modelled as an atomic concept, then it would not be possible to infer that persons who are citizens *only* of Italy cannot have dual-nationality (i.e., cannot be citizens of more than one country).

Finally, nominals can be viewed as a powerful generalisation of DL *Abox individuals* [Schaerf, 1994]: in an Abox we can assert that an individual is an instance of a concept or that a pair of individuals is an instance of a role (binary relation), but Abox individuals cannot be used *inside* concepts. For example, if Giuseppe and Italy are Abox individuals, we could assert that the pair (Giuseppe, Italy) is an instance of the *citizen-of* role, but we could not describe the concept Italian as a Person who is a *citizen-of* Italy. Using nominals, not only can we express this concept (i.e., $\text{Person} \sqcap \exists \text{citizen-of}.\text{Italy}$), but we can also capture Abox assertions with concept inclusion axioms of the form $\text{Giuseppe} \sqsubseteq \text{Italian}$ (Giuseppe is an Italian) and $\text{Giuseppe} \sqsubseteq \exists \text{citizen-of}.\text{Italy}$ (Giuseppe is a *citizen-of* Italy).