Reasoning with Expressive Description Logics: Theory and Practice

Ian Horrocks

Department of Computer Science, University of Manchester
Oxford Road, Manchester M13 9PL, UK
horrocks@cs.man.ac.uk

Abstract. Description Logics are a family of class based knowledge representation formalisms characterised by the use of various constructors to build complex classes from simpler ones, and by an emphasis on the provision of sound, complete and (empirically) tractable reasoning services. They have a wide range of applications, but their use as ontology languages has been highlighted by the recent explosion of interest in the "Semantic Web", where ontologies are set to play a key role. DAML+OIL is a description logic based ontology language specifically designed for use on the Web. The logical basis of the language means that reasoning services can be provided, both to support ontology design and to make DAML+OIL described Web resources more accessible to automated processes.

1 Introduction

Description Logics (DLs) are a family of class (concept) based knowledge representation formalisms. They are characterised by the use of various constructors to build complex concepts from simpler ones, an emphasis on the decidability of key reasoning tasks, and by the provision of sound, complete and (empirically) tractable reasoning services.

Description logics have been used in a range of applications, e.g., configuration [42], and reasoning with database schemas and queries [14, 12, 36]. They are also widely used as a formal basis for ontology languages and to provide reasoning support for ontology design and deployment. This latter application has been highlighted by the recent explosion of interest in the "Semantic Web", where ontologies are set to play a key role [30].

The current Web consists mainly of handwritten and machine generated HTML pages primarily intended for direct human processing (reading, browsing, form-filling, etc.). The aim of the so called semantic web is to make Web resources (not just HTML pages, but a wide range of web accessible services) more readily accessible to automated processes by adding meta-data annotations that describe their content [7]. Ontologies will be used as a a source of shared and precisely defined terms that can be used in such meta-data.

1.1 Ontologies

An ontology typically consists of a hierarchical description of important concepts (classes) in a domain, along with descriptions of the properties of each concept. The degree of formality employed in capturing these descriptions can be quite variable, ranging from natural language to logical formalisms, but increased formality and regularity clearly facilitates machine processing.

Examples of the use of ontologies could include:

- in e-commerce sites [39], where ontologies can facilitate machine-based communication between buyer and seller, enable vertical integration of markets (see, e.g., http://www.verticalnet.com/), and allow descriptions to be reused in different marketplaces;
- in search engines [40], where ontologies can help searching to go beyond the current keyword-based approach, and allow pages to be found that contain syntactically different, but semantically similar words/phrases (see, e.g., http://www.hotbot.com/);
- in Web services [43], where ontologies can provide semantically richer service descriptions that can be more flexibly interpreted by intelligent agents.

2 Ontology Languages

The recognition of the key role that ontologies are likely to play in the future of the Web 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,[1] RDF[2] (Resource Description Framework), and RDF Schema [16]. 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.

RDFS is, however, 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, e.g., to determine the semantic relationship between syntactically different terms.

A recognition of the limitations of RDFS led to the development of new web ontology languages, in particular OIL [19, 20], a language developed by a group of (largely) European researchers, several of whom were members of the On-To-Knowledge consortium,[3] and DAML-ONT [25], a language developed in the DARPA Agent Markup Language (DAML) program.[4] These two languages were

[1] http://www.w3.org/XML/Schema/
[2] http://www.w3c.org/RDF/
[3] http://www.ontoknowledge.org/oil
[4] http://www.daml.org/

subsequently merged to produce DAML+OIL, which has recently been submitted to W3C[5] and forms the basis of a proposed W3C Web ontology language.[6].

2.1 DAML+OIL

DAML+OIL describes the structure of a domain in terms of *classes* and *properties*. A DAML+OIL ontology consists of a set of *axioms* that assert, e.g., subsumption relationships between classes or properties. Instances of classes (properties) are assumed to be RDF resources[7] (pairs of RDF resources). Asserting that a given resource (pair of resources) is an instance of a given DAML+OIL class (property) is left to RDF, a task for which it is well suited.

From a formal point of view, DAML+OIL can be seen to be equivalent to a very expressive description logic, with a DAML+OIL ontology corresponding to the Tbox, and RDF type and property assertions corresponding to the Abox. As in a DL, DAML+OIL classes can be names (URIs) or *expressions*, and a variety of *constructors* are provided for building class expressions; the expressive power of the language is determined by the class (and property) constructors supported, and by the kinds of axiom supported.

Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	$\neg$Male
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}
toClass	$\forall P.C$	$\forall$hasChild.Doctor
hasClass	$\exists P.C$	$\exists$hasChild.Lawyer
hasValue	$\exists P.\{x\}$	$\exists$citizenOf.{USA}
minCardinalityQ	$\geqslant nP.C$	$\geqslant$2hasChild.Lawyer
maxCardinalityQ	$\leqslant nP.C$	$\leqslant$1hasChild.Male
cardinalityQ	$=n\,P.C$	=1 hasParent.Female

Fig. 1. DAML+OIL class constructors

Figure 1 summarises the constructors supported by DAML+OIL, where C (possibly subscripted) is a class, P is a property, x is an individual and n is a non-negative integer. The standard DL syntax is used for compactness as the RDF syntax is rather verbose. In the RDF syntax, for example, Human $\sqcap$ Male would be written as

[5] http://www.w3.org/Submission/2001/12/

[6] http://www.w3c.org/2001/sw/WebOnt/

[7] Everything describable by RDF is called a resource. A resource could be Web accessible, e.g., a Web page or part of a Web page, but it could also be an object that is not directly accessible via the Web, e.g., a person. Resources are named by URIs plus optional anchor ids. See http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ for more details.

```
<daml:Class>
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Human"/>
    <daml:Class rdf:about="#Male"/>
  </daml:intersectionOf>
</daml:Class>
```

while $\geqslant$2hasChild.Lawyer would be written as

```
<daml:Restriction daml:minCardinalityQ="2">
  <daml:onProperty rdf:resource="#hasChild"/>
  <daml:hasClassQ rdf:resource="#Lawyer"/>
</daml:Restriction>
```

DAML+OIL also supports the use of XML Schema *datatypes* in class expressions. These can be so called primitive datatypes, such as strings, decimal or float, as well as more complex derived datatypes such as integer sub-ranges. Datatypes can be used instead of classes in `toClass` and `hasClass` constructs (e.g., `hasClass`ageinteger), and data values can be used in the `hasValue` construct (e.g., `hasValue`age(integer21)).

The meaning of the language is defined by DAML+OIL's model-theoretic semantics.[8]. The semantics is based on interpretations, where an interpretation consists of a domain of discourse and an interpretation function. The domain is divided into two disjoint sets, the "object domain" $\Delta^{\mathcal{I}}$ and the "datatype domain" Δ_D. The interpretation function $\mathcal{I}$ maps classes into subsets of the object domain, individuals into elements of the object domain, datatypes into subsets of the datatype domain and data values into elements of the datatype domain. In addition, two disjoint sets of properties are distinguished: object properties and datatype properties. The interpretation function maps the former into subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and the latter into subsets of $\Delta^{\mathcal{I}} \times \Delta_D$.

Figure 2 summarises the axioms supported by DAML+OIL, where C (possibly subscripted) is a class, P (possibly subscripted) is a property, P^- is the inverse of P, P^+ is the transitive closure of P, x (possibly subscripted) is an individual and $\top$ is an abbreviation for $A \sqcup \neg A$ for some class A. These axioms make it possible to assert subsumption or equivalence with respect to classes or properties, the disjointness of classes, the equivalence or non-equivalence of individuals, and various properties of properties.

A crucial feature of DAML+OIL is that subClassOf and sameClassAs axioms can be applied to arbitrary class expressions. This provides greatly increased expressive power with respect to standard frame-based languages where such axioms are invariably restricted to so called *definitions*, where the left hand side is an atomic name, there is only one such axiom per name, and there are no definitional cycles (the class on the right hand side of an axiom cannot refer, either directly or indirectly, to the class name on the left hand side).

[8] http://www.w3.org/TR/daml+oil-model

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
sameClassAs	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost $\equiv$ price
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
uniqueProperty	$\top \sqsubseteq \leqslant 1P$	$\top \sqsubseteq \leqslant$1hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1P^-$	$\top \sqsubseteq \leqslant$1isMotherOf$^-$

Fig. 2. DAML+OIL axioms

A consequence of the expressive power of DAML+OIL is that all of the class and individual axioms, as well as the uniqueProperty and unambiguousProperty axioms, can be reduced to subClassOf and sameClassAs axioms (as can be seen from the DL syntax). In fact sameClassAs could also be reduced to subClassOf as a sameClassAs axiom $C \equiv D$ is trivially equivalent to a pair of subClassOf axioms, $C \sqsubseteq D$ and $D \sqsubseteq C$. Moreover, the distinction between Tbox and Abox breaks down, as Abox assertions can be expressed using Tbox axioms. E.g., $x \in C$ can be expressed as $\{x\} \sqsubseteq C$, and $\langle x, y\rangle \in P$ can be expressed as $\{x\} \sqsubseteq \exists P.\{y\}$.

As far as property axioms are concerned, it is possible to assert that a given property is unique (functional), unambiguous (inverse functional) or transitive (i.e., that its interpretation must be closed under composition). It is also possible to assign a name to the inverse of a property, thus allowing inverse properties to be used in class expressions. Transitive properties are preferred over transitive closure as this has been shown to facilitate the design of (efficient) algorithms [33].

As usual, an interpretation is called a model of an ontology $\mathcal{O}$ if it satisfies each of the axioms in $\mathcal{O}$. An ontology $\mathcal{O}$ is said to be satisfiable if it has a model, and a class C is said to be satisfiable w.r.t. $\mathcal{O}$ if there is a model of $\mathcal{O}$ in which the interpretation of C is non-empty.

3 Inference Problems

As we have seen, DAML+OIL is equivalent to a very expressive description logic. More precisely, DAML+OIL is equivalent to the $\mathcal{SHIQ}$ DL [32] with the addition of existentially defined classes (i.e., the oneOf constructor) and *datatypes* (often called concrete domains in DLs [3]). This equivalence allows DAML+OIL to 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 [18];

– as a source of sound and complete algorithms and optimised implementation techniques for deciding key inference problems [32, 31];
– to use implemented DL systems in order to provide (partial) reasoning support [27, 46, 23].

Key inference problems (w.r.t. an ontology $\mathcal{O}$) include:

Consistency Check if the knowledge is meaningful.
- Is $\mathcal{O}$ consistent? i.e., does there exists a model $\mathcal{I}$ of $\mathcal{O}$?
- Is C consistent? i.e., $C^{\mathcal{I}} \neq \emptyset$ in some model $\mathcal{I}$ of $\mathcal{O}$?

Subsumption Structure knowledge and compute a taxonomy.
- $C \sqsubseteq D$ w.r.t. $\mathcal{O}$? i.e., does $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ hold in all models $\mathcal{I}$ of $\mathcal{O}$?

Equivalence Check if two classes denote same set of instances.
- $C \equiv D$ w.r.t. $\mathcal{O}$? i.e., does $C^{\mathcal{I}} = D^{\mathcal{I}}$ hold in all models $\mathcal{I}$ of $\mathcal{O}$?

Instantiation Check if individual is an instance of a class.
- $i \in C$ w.r.t. $\mathcal{O}$? i.e., does $i^{\mathcal{I}} \in C^{\mathcal{I}}$ hold in all models $\mathcal{I}$ of $\mathcal{O}$?

Retrieval Retrieve the set of individuals that instantiate a class.
- Retrieve the set of i s.t. $i \in C$ w.r.t. $\mathcal{O}$.

In DAML+OIL, all of the above problems are recucible to class consistency (satisfiability) w.r.t. an ontology. In particular, $C \sqsubseteq D$ w.r.t. $\mathcal{O}$ (written $C \sqsubseteq_{\mathcal{O}} D$) iff $D \sqcap \neg C$ is not consistent w.r.t. $\mathcal{O}$, and $i \in C$ w.r.t. $\mathcal{O}$ (written $i \in_{\mathcal{O}} C$) iff $\{i\} \sqcap \neg C$ is not consistent w.r.t. $\mathcal{O}$. Moreover, by using a rewriting technique called *internalisation*, satisfiability w.r.t. an ontology can be reduced to the problem of determining the satisfiability of a single concept [32].

Reasoning can be useful at many stages during the design, maintenance and deployment of ontologies.

– Reasoning can be used to support ontology design and to improve the quality of the resulting ontology. For example, class consistency and subsumption reasoning can be used to check for logically inconsistent classes and (possibly unexpected) implicit subsumption relationships [6]. This kind of support has been shown to be particularly important with large ontologies, which are often built and maintained over a long period by multiple authors. Other "non-standard" reasoning tasks, such as approximation, matching, unification and computing least common subsumers could also be used to support "bottom up" ontology design, i.e., the identification and description of relevant classes from sets of example instances [10].
– Like information integration [15], ontology integration can also be supported by reasoning. For example, integration can be performed using inter-ontology assertions specifying relationships between classes and properties, with reasoning being used to compute the integrated hierarchy and to highlight any problems/inconsistencies. Unlike some other integration techniques (e.g., name reconciliation [41]), this method has the advantage of being non-intrusive with respect to the original ontologies.

– Reasoning with respect to deployed ontologies will enhance the power of "intelligent agents", allowing them to determine if a set of facts is consistent w.r.t. an ontology, to identify individuals that are implicitly members of a given class etc. A suitable service ontology could, for example, allow an agent seeking secure services to identify a service requiring a userid and password as a possible candidate.

4 Reasoning with Datatypes

DAML+OIL supports arbitrary XML Schema datatypes. This is facilitated by maintaining a clean separation between instances of "object" classes (defined using the ontology language) and instances of datatypes (defined using the XML Schema type system). In particular, as mentioned in Section 2.1, it is assumed that that the domain of interpretation of object classes is disjoint from the domain of interpretation of datatypes, so that an instance of an object class (e.g., the individual "Italy") can never have the same interpretation as a value of a datatype (e.g., the integer 5), and that the set of object properties (which map individuals to individuals) is disjoint from the set of datatype properties (which map individuals to datatype values).

The disjointness of object and datatype domains and properties was motivated by both philosophical and pragmatic considerations:

– Datatypes are considered to be already sufficiently structured by the built-in predicates, and it is, therefore, not appropriate to form new classes of datatype values using the ontology language [26].
– The simplicity and compactness of the ontology language are not compromised: even enumerating all the XML Schema datatypes would add greatly to its complexity, while adding a logical theory for each datatype, even if it were possible, would lead to a language of monumental proportions.
– The semantic integrity of the language is not compromised—defining theories for all the XML Schema datatypes would be difficult or impossible without extending the language in directions whose semantics may be difficult to capture within the existing framework.
– The "implementability" of the language is not compromised—a *hybrid* reasoner can easily be implemented by combining a reasoner for the "object" language with one capable of deciding satisfiability questions with respect to conjunctions of (possibly negated) datatypes [31].

From a theoretical point of view, this design means that the ontology language can specify constraints on data values, but as data values can never be instances of object classes they cannot apply additional constraints to elements of the object domain. This allows the type system to be extended without having any impact on the ontology language, and vice versa. Similarly, the formal properties of hybrid reasoners are determined by those of the two components; in particular, the combined reasoner will be sound and complete if both components are sound and complete.

From a practical point of view, DAML+OIL implementations will probably choose to support only a subset of the available XML Schema datatypes. For supported data types, they can either implement their own type checker/validater or rely on some external component. The job of a type checker/validater is simply to take zero or more (possibly negated) data values and one or more (possibly negated) datatypes, and determine if there exists an element of Δ_D that is (not) equal to the interpretation of every one of the (negated) data values and (not) in the interpretation of every one of the (negated) data types.

5 Practical Reasoning Services

The concept satisfiability problem for expressive DLs is known to be of high complexity: at least PSPACE-complete, and rising to EXPTIME-complete for very expressive logics such as $\mathcal{SHIQ}$ [17]. Fortunately, the pathological cases that lead to such high *worst case* complexity are rather artificial, and rarely occur in practice [44, 24, 49, 28]; by employing a wide range of optimisations it has proved possible to implement systems that exhibit good *typical case* performance in realistic applications [2, 11, 28, 23, 47].

Most modern DL systems use *tableaux* algorithms to test concept satisfiability. These algorithms work by trying to construct (a tree representation of) a model of the concept, starting from an individual instance. Tableaux expansion rules decompose concept expressions, add new individuals (e.g., as required by $\exists R.C$ terms),[9] and merge existing individuals (e.g., as reqired by $\leqslant nR.C$ terms). Nondeterminism (e.g., resulting from the expansion of disjunctions) is dealt with by searching the various possible models. For an unsatisfiable concept, all possible expansions will lead to the discovery of an obvious contradiction known as a *clash* (e.g., an individual that must be an instance of both A and $\neg A$ for some concept A), and for a satisfiable concept, a complete and clash-free model will be constructed [33].

Tableaux algorithms have many advantages. It is relatively easy to design provably sound, complete and terminating algorithms, and the basic technique can be extended to deal with a wide range of class and role constructors. Moreover, although many algorithms have a higher worst case complexity than that of the underlying problem, they are usually quite efficient at solving the relatively easy problems that are typical of realistic applications.

Even in realistic applications, however, problems can occur that are much too hard to be solved by naive implementations of theoretical algorithms. Modern DL systems, therefore, include a wide range of optimisation techniques, the use of which has been shown to improve typical case performance by several orders of magnitude [29]. These techniques include lazy unfolding, absorption and dependency directed backtracking.

[9] Cycle detection techniques known as *blocking* may be required in order to guarantee termination.

5.1 Lazy Unfolding

In an ontology, or DL Tbox, large and complex concepts are seldom described monolithically, but are built up from a hierarchy of named concepts whose descriptions are less complex. The tableaux algorithm can take advantage of this structure by trying to find contradictions between concept names before adding expressions derived from Tbox axioms. This strategy is known as *lazy unfolding* [1, 28].

The benefits of lazy unfolding can be maximised by lexically *normalising* and *naming* all concept expressions and, recursively, their sub-expressions. An expression C is normalised by rewritting it in a standard form (e.g., disjunctions are rewritten as negated conjunctions), and named by substituting it with a new concept name A, with an axiom $A \equiv C$ being added to the Tbox. The normalisation step allows lexically equivalent expressions to be recognised and identically named, and can even detect syntactiacally "obvious" satisfiability and unsatisfiability.

5.2 Absorption

Not all axioms are amenable to lazy unfolding. In particular, so called *general concept incusions* (GCIs), axioms of the form $C \sqsubseteq D$ where C is non-atomic, must be dealt with by explicitly making every individual in the model an instance of $D \sqcup \neg C$. Large numbers of GCIs result in a very high degree of non-determinism and catastrophic performance degredation [28].

Absorption is another rewriting technique that tries to reduce the number of GCIs in the Tbox by absorbing them into axioms of the form $A \sqsubseteq C$, where A is a concept name. The basic idea is that an axiom of the form $A \sqcap D \sqsubseteq D'$ can be rewritten as $A \sqsubseteq D' \sqcup \neg D$ and absorbed into an existing $A \sqsubseteq C$ axiom to give $A \sqsubseteq C \sqcap (D' \sqcup \neg D)$ [37]. Although the disjunction is still present, lazy unfolding ensures that it is only applied to individuals that are already known to be instances of A.

5.3 Dependency Directed Backtracking

Inherent unsatisfiability concealed in sub-expressions can lead to large amounts of unproductive backtracking search known as thrashing. For example, expanding the expression $(C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n) \sqcap \exists R.(A \sqcap B) \sqcap \forall R.\neg A$ could lead to the fruitless exploration of 2^n possible expansions of $(C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)$ before the inherent unsatisfiability of $\exists R.(A \sqcap B) \sqcap \forall R.\neg A$ is discovered. This problem is addressed by adapting a form of dependency directed backtracking called *backjumping*, which has been used in solving constraint satisfiability problems [5].

Backjumping works by labeling concepts with a dependency set indicating the non-deterministic expansion choices on which they depend. When a clash is discovered, the dependency sets of the clashing concepts can be used to identify the most recent non-deterministic expansion where an alternative choice might

alleviate the cause of the clash. The algorithm can then jump back over intervening non-deterministic expansions *without* exploring any alternative choices. Similar techinques have been used in first order theorem provers, e.g., the "proof condensation" technique employed in the HARP theorem prover [45].

6 Research Challenges for DAML+OIL

Class consistency/subsumption reasoning in DAML+OIL is know to be decidable (as it is contained in the C2 fragment of first order logic [21]), but many challenges remain for implementors of "practical" reasoning systems, i.e., systems that perform well with the kinds of reasoning problem generated by realistic applications.

6.1 Individuals

The OIL language was designed so that it could be mapped to the $\mathcal{SHIQ}$ DL, thereby providing a implementation path for reasoning services. This mapping is made possible by a very weak treatment of individuals occurring in existentially defined classes, which are treated not as single elements but as the extensions of corresponding primitive classes. This is a well known technique for avoiding the reasoning problems that arise with existentially defined classes (such as classes defined using DAML+OIL's oneOf constructor) and is also used, e.g., in the CLASSIC knowledge representation system [8].

In contrast, DAML+OIL gives a standard semantics to such individuals, i.e., they are interpreted as single elements in the domain of discourse. This treatment of individuals is very powerful, and justifies intuitive inferences that would not be valid for OIL, e.g., that persons all of whose countries of residence are (oneOf) Italy are kinds of person that have at most one country of residence:

$$\text{Person} \sqcap \forall \text{residence}.\{\text{Italy}\} \sqsubseteq \leqslant 1\text{residence}$$

Unfortunately, the combination of such individuals with inverse properties is so powerful that it pushes the worst case complexity of the class consistency problem from EXPTIME (for $\mathcal{SHIQ}$/OIL) to NEXPTIME [31]. No "practical" decision procedure is currently known for this logic, and there is no implemented system that can provide sound and complete reasoning for the whole DAML+OIL language. In the absence of inverse properties, however, a tableaux algorithm has been devised [31], and in the absence of individuals (in existentially defined classes), DAML+OIL can exploit implemented DL systems via a translation into $\mathcal{SHIQ}$ (extended with datatypes) similar to the one used by OIL. It would, of course, also be possible to translate DAML+OIL ontologies into $\mathcal{SHIQ}$ using OIL's weak treatment of individuals,[10] but in this case reasoning with individuals would not be sound and complete with respect to the semantics of the language.

[10] This approach is taken by some existing applications, e.g., OilEd [6].

6.2 Scalability

Even without the oneOf constructor, class consistency reasoning is still a hard problem. Moreover, Web ontologies can be expected to grow very large, and with deployed ontologies it may also be desirable to reason w.r.t. a large numbers of class/property instances.

There is good evidence of empirical tractability and scalability for implemented DL systems [28, 22], but this is mostly w.r.t. logics that do not include inverse properties (e.g., $\mathcal{SHF}$ [11]). Adding inverse properties makes practical implementations more problematical as several important optimisation techniques become much less effective. Work is required in order to develop more highly optimised implementations supporting inverse properties, and to demonstrate that they can scale as well as $\mathcal{SHF}$ implementations. It is also unclear if existing techniques will be able to cope with large numbers of class/property instances [34].

Finally, it is an inevitable consequence of the high worst case complexity that some problems will be intractable, even for highly optimised implementations. It is conjectured that such problems rarely arise in practice, but the evidence for this conjecture is drawn from a relatively small number of applications, and it remains to be seen if a much wider range of Web application domains will demonstrate similar characteristics.

6.3 New Reasoning Tasks

So far we have mainly discussed class consistency/subsumption reasoning, but this may not be the only reasoning problem that is of interest. Other tasks could include querying, explanation, approximation, matching, unification, computing least common subsumers, etc. Querying in particular may be important in Semantic Web applications. Some work on query languages for description logics has already been done [48, 13, 35], and work is underway on the design of a DAML+OIL query language, but the computational properties of such a language, either theoretical or empirical, have yet to be determined.

Explanation may also be an important problem, e.g., to help an ontology designer to rectify problems identified by reasoning support, or to explain to a user why an application behaved in an unexpected manner. As discussed in Section 3, so called "non-standard inferences" such as approximation, matching, unification and computing least common subsumers could also be important in ontology design. Non-standard inferences are the subject of ongoing research [4, 10, 38, 9], but it is not clear if they can be extended to deal with languages as expressive as DAML+OIL.

7 Summary

Description Logics are a family of class based knowledge representation formalisms characterised by the use of various constructors to build complex classes

[11] $\mathcal{SHF}$ is equivalent to $\mathcal{SHIQ}$ without inverse properties and with only functional properties instead of qualified number restrictions [32].

from simpler ones, and by an emphasis on the provision of sound, complete and (empirically) tractable reasoning services. They have been used in a wide range of applications, but perhaps most notably (at least in recent times) in providing a formal basis and reasoning support for ontology languages such as DAML+OIL.

DAML+OIL can be seen as a very expressive DL with an XML/RDF syntax. One of the key advantages of basing the language on a DL is that it is possible to provide reasoning services, both to support ontology design and to make DAML+OIL described Web resources more accessible to automated processes. This combination of features has proved very popular: DAML+OIL is already being widely used, and some major efforts are committed to encoding their ontologies in the language. This has been particularly evident in the bio-ontology domain, where the Bio-Ontology Consortium has specified DAML+OIL as their ontology exchange language, and the Gene Ontology [50] is being migrated to DAML+OIL in a project partially funded by GlaxoSmithKline Pharmaceuticals in cooperation with the Gene Ontology Consortium.[12]

What of the future? The development of the semantic Web, and of Web ontology languages, presents many opportunities and challenges. Even for less expressive languages, acceptable performance can only be achieved by using a wide range of optimisation techniques. A "practical" (satisfiability/subsumption) algorithm for the full DAML+OIL language has yet to be developed, and it is not yet clear that sound and complete reasoners will be able to provide adequate performance for typical Web applications. Finding answers to these questions is the subject of ongoing research.

Acknowledgements

I would like to acknowledge the contribution of all those involved in the development of DAML-ONT, OIL and DAML+OIL, amongst whom Dieter Fensel, Frank van Harmelen, Deborah McGuinness and Peter F. Patel-Schneider deserve particular mention, as well as the contribution made by members of Franz Baader's group at TU Dresden, in particular Ulrike Sattler and Stephan Tobies.

References

1. F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems, or: Making KRIS get a move on. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 270–281, 1992.
2. F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.
3. F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 452–457, 1991.

[12] http://www.geneontology.org/.

4. F. Baader, R. Küsters, A. Borgida, and D. L. McGuinness. Matching in description logics. *J. of Logic and Computation*, 9(3):411–447, 1999.
5. A. B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results*. PhD thesis, University of Oregon, 1995.
6. S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reason-able ontology editor for the semantic web. In *Proc. of the Joint German/Austrian Conf. on Artificial Intelligence (KI 2001)*, number 2174 in Lecture Notes in Artificial Intelligence, pages 396–408. Springer-Verlag, 2001.
7. T. Berners-Lee. *Weaving the Web*. Harpur, San Francisco, 1999.
8. A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.
9. S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In *Proc. of the 8th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'2002)*, pages 203–214, 2002.
10. S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics — what does it buy me? In *Proc. of KI-2001 Workshop on Applications of Description Logics (KIDLWS'01)*, volume 44 of *CEUR (http://ceur-ws.org/)*, 2001.
11. P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: Preliminary report. In *Proc. of the 1995 Description Logic Workshop (DL'95)*, pages 131–139, 1995.
12. D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.
13. D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views in description logics. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 9–13. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-22/, 1999.
14. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.
15. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.
16. S. Decker, F. van Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein, and S. Melnik. The semantic web: The roles of XML and RDF. *IEEE Internet Computing*, 4(5), 2000.
17. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91)*, pages 151–162. Morgan Kaufmann, Los Altos, 1991.
18. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.
19. D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In R. Dieng, editor, *Proc. of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW'00)*, number 1937 in Lecture Notes in Artificial Intelligence, pages 1–16. Springer-Verlag, 2000.
20. D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

21. E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *Proc. of the 12th IEEE Symp. on Logic in Computer Science (LICS'97)*, pages 306–317. IEEE Computer Society Press, 1997.
22. V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, 2001.
23. V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.
24. J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 68:367–397, 1994.
25. J. Hendler and D. L. McGuinness. The darpa agent markup language". *IEEE Intelligent Systems*, 15(6):67–73, 2000.
26. B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91)*, pages 335–346, 1991.
27. I. Horrocks. The FaCT system. In H. de Swart, editor, *Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 307–312. Springer-Verlag, 1998.
28. I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.
29. I. Horrocks and P. F. Patel-Schneider. Optimizing description logic subsumption. *J. of Logic and Computation*, 9(3):267–293, 1999.
30. 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. To appear.
31. I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*. Morgan Kaufmann, Los Altos, 2001.
32. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. 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-Verlag, 1999.
33. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.
34. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, number 1831 in Lecture Notes in Artificial Intelligence, pages 482–496. Springer-Verlag, 2000.
35. I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 399–404, 2000.
36. I. Horrocks, S. Tessaris, U. Sattler, and S. Tobies. How to decide query containment under constraints using a description logic. In *Proc. of the 7th Int. Workshop on Knowledge Representation meets Databases (KRDB 2000)*. CEUR (http://ceur-ws.org/), 2000.
37. I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 285–296, 2000.

38. R. Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2001.
39. D. L. McGuinness. Ontological issues for knowledge-enhanced search. In *Proc. of FOIS*, Frontiers in Artificial Intelligence and Applications. IOS-press, 1998.
40. D. L. McGuinness. Ontologies for electronic commerce. In *Proc. of the AAAI '99 Artificial Intelligence for Electronic Commerce Workshop*, 1999.
41. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. The Chimaera ontology environment. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, 2000.
42. D. L. McGuinness and J. R. Wright. An industrial strength description logic-based configuration platform. *IEEE Intelligent Systems*, pages 69–77, 1998.
43. S. McIlraith, T. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, March/April 2001.
44. B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.
45. F. Oppacher and E. Suen. HARP: A tableau-based theorem prover. *J. of Automated Reasoning*, 4:69–100, 1988.
46. P. F. Patel-Schneider. DLP system description. In *Proc. of the 1998 Description Logic Workshop (DL'98)*, pages 87–89. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-11/, 1998.
47. P. F. Patel-Schneider. DLP. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 9–13. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-22/, 1999.
48. M.-C. Rousset. Backward reasoning in ABoxes for query answering. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 18–22. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-22/, 1999.
49. P.-H. Speel, F. van Raalte, P. E. van der Vet, and N. J. I. Mars. Runtime and memory usage performance of description logics. In G. Ellis, R. A. Levinson, A. Fall, and V. Dahl, editors, *Knowledge Retrieval, Use and Storage for Efficiency: Proc. of the 1st Int. KRUSE Symposium*, pages 13–27, 1995.
50. The Gene Ontology Consortium. Gene ontolgy: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.