OilEd: a Reason-able Ontology Editor for the Semantic Web

Sean Bechhofer, Ian Horrocks, Carole Goble and Robert Stevens
Department of Computer Science,
University of Manchester, UK
seanb@cs.man.ac.uk,
http://img.cs.man.ac.uk

Abstract

Ontologies will play a pivotal rôle in the "Semantic Web", where they will provide a source of precisely defined terms that can be communicated across people and applications. OilEd, is an ontology editor that has an easy to use frame interface, yet at the same time allows users to exploit the full power of an expressive web ontology language (OIL). OilEd uses reasoning to support ontology design, facilitating the development of ontologies that are both more detailed and more accurate.

1 Introduction

Ontologies have become an increasingly important research topic. This is a result both of their usefulness in a range of application domains [1, 2, 3], and of the pivotal rôle that they are set to play in the development of the *Semantic Web*. The Semantic Web vision, as articulated by Tim Berners-Lee [4], is of a Web in which resources are accessible not only to humans, but also to automated processes, e.g., automated "agents" roaming the web performing useful tasks such as improved search (in terms of precision) and resource discovery, information brokering and information filtering. The automation of tasks depends on elevating the status of the web from machine-readable to something we might call machine-understandable. The key idea is to have data on the web defined and linked in such a way that its meaning is explicitly interpretable by software processes rather than just being implicitly interpretable by humans.

OIL [5] is a language that has been developed for the representation of ontologies within the Semantic Web. It extends RDF Schema (RDFS) [6] – itself a proposed ontology/knowledge representation language – with a much richer set of modelling primitives. A similar RDFS based web ontology language called DAML has been developed as part of the DARPA DAML project [7]. These two languages have now been merged under the name DAML+OIL, although there are some differences between the approaches used in the languages [8].

OIL has a frame-like syntax, which facilitates tool building, yet can be mapped onto an expressive description logic (DL), which facilitates the provision of reasoning services. OilEd is an ontology editing tool for OIL (and DAML+OIL) that exploits both these features in order to provide a familiar and intuitive style of user interface with the added benefit of reasoning support. Its main novelty lies in the extension of the frame editor paradigm to deal with a very expressive language, and the use of a highly optimised DL reasoning engine to provide sound and complete yet still empirically tractable reasoning services.

Reasoning with terms from deployed ontologies will be important for the Semantic Web, but reasoning support is also extremely valuable at the ontology design phase, where it can be used to to detect logically inconsistent classes and to discover implicit subclass relations. This encourages a more descriptive approach to ontology design, with the reasoner being used to infer part of the subsumption lattice; the resulting ontologies contain fewer errors, yet provide more detailed descriptions that can be exploited by automated processes in the Semantic Web. Finally, reasoning is of particular benefit when ontologies are large and/or multiply authored, and also facilitates ontology sharing, merging and integration [9]; considerations that will be particularly important in the distributed web environment.

2 Oil and DAML+OIL

The development of OIL resulted from efforts to combine the best features of frame and DL based knowledge representation systems, while at the same time maximising compatibility with emerging web standards. The intention was to design a language that was intuitive to human users, and yet provided adequate expressive power for realistic applications (many early DLs failed on this second count—see [10]).

The resulting language combines a familiar frame like syntax (derived in part from the OKBC-lite knowledge model [11]), with the power and flexibility of a DL (i.e., boolean connectives, unlimited nesting of class elements, transitive and inverse slots, general axioms, etc.). The language is defined as an extension of RDFS, thereby making OIL ontologies (partially) accessible to any "RDFS-aware" application.

The frame syntax is less daunting to ontologists/domain experts than a DL style syntax, and it facilitates a modelling style in which ontologies start out simple (in terms of their descriptive content) and are gradually extended, both as the design itself is refined and as users become more familiar with the language's advanced features. The frame paradigm also facilitates the construction and

slot-def part-of
  subslot-of structural-relation
  inverse has-part
  properties *transitive*
	class-def *defined* herbivore
	  subclass-of animal
	  slot-constraint eats
	    value-type plant *OR*
	      slot-constraint part-of
	        has-value plant

Figure 1: OIL language example

adaption of tools, e.g., the OntoEdit and Protégé editors and the Chimaera integration tool are all being adapted to use OIL/DAML+OIL [12, 13, 9].

On the other hand, basing the language on an underlying mapping to a very expressive DL ($\mathcal{SHIQ}$) provides a well defined semantics and a clear understanding of its formal properties. In particular the class subsumption/satisfiability problem is decidable and has worst case ExpTime complexity [14]. The mapping also provides a mechanism for the provision of practical reasoning services by exploiting implemented DL systems, e.g., the FaCT system [15].

OIL extends standard frame languages in a number of directions. One of the key ideas is that an anonymous class description, or even boolean combinations of class descriptions, can occur anywhere that a class name would ordinarily be used, e.g., in slot constraints and in the list of superclasses. For example, in Figure 1, a herbivore is described as an animal that eats only plants or part-of plants. Points to note are that universally quantified (value-type) and existentially quantified (has-value) slot constraints are clearly differentiated, and that the constraint on the eats slot is a disjunction, one of whose components is an anonymous class description (in this case, just a single slot constraint). In addition, it is asserted that the part-of slot is transitive, and that its inverse is the slot has-part. A complete specification of the language can be found in [5].

3 OilEd

OilEd is a simple ontology editor that supports the construction of OIL-based ontologies. The basic design has been heavily influenced by similar tools such as Protégé [13] and OntoEdit [12], but OilEd extends these approaches in a number of ways, notably through an extension of expressive power and the use of a reasoner.

However, OilEd is not intended as a replacement for such tools—the current implementation of OilEd is intended primarily as a prototype to test and demonstrate novel ideas, and compromises have been made in the design and implementation. For example, the tool does not provide key functionality for

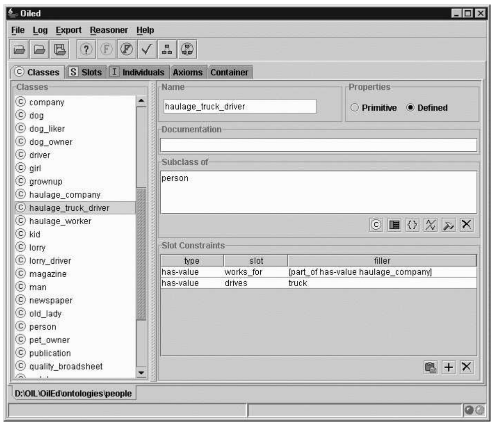

Figure 2: OilEd Class Panel

collaborative ontology development such as versioning, integration and merging of ontologies. Similarly, the powerful tailorability and knowledge acquisition aspects of tools such as Protégé have been ignored completely. Rather, the design has concentrated on demonstrating how the frame paradigm can be extended to deal with a more expressive modelling language, and how reasoning can be used to support the design and maintenance of ontologies.

The central component used throughout OilEd is the notion of a *frame description*. This consists of a collection of superclasses along with a list of slot constraints. This is similar to other frame systems. Where OilEd differs, however, is that wherever a class name can appear, an anonymous frame description or arbitrary complexity can be used. In addition, boolean combinations of frames or classes (using **and**, **or** and **not**) can also appear. This is in contrast to conventional frame systems, where in general, slot constraints and superclasses must be class names.

As well as being able to assert individuals as slot fillers, several types of constraints on slot fillers can be asserted (these kinds of constraint are sometimes called *facets*). These include *value-type* (universal) restrictions, *has-value* (existential) restrictions, and explicit *cardinality* restrictions (e.g., at most three fillers of a given class). Each constraint has a clearly defined meaning, removing the confusion present in some frame systems, where, for example, it is not always clear whether the semantics of a slot-constraint should be interpreted as a universal or existential quantification.

A *class definition* specifies the class name, along with an optional frame description (see above) and a specification of whether the class is *defined* or *primitive*. If defined, the class is taken to be equivalent to the given description (necessary and sufficient conditions). If primitive, the class is taken to be an explicit subclass of the given description (necessary conditions). In the specification of the OIL language, classes can have multiple definitions. In OilEd, this is not allowed—classes must have a single definition—but the same effect can be achieved through the use of *equivalence* axioms as discussed below.

A *slot definition* gives the name of the slot and allows additional properties of the slot to be asserted, e.g., the names of any *superslots* or *inverses*. Domain and range restrictions on a slot can be specified. For example, we can constrain the relationship parent to have both domain and range person, asserting that only persons can have, and be, parents. As with class descriptions, the domain and range restrictions can be arbitrary class expressions such as anonymous frames or boolean combinations of classes or frames, again extending the expressivity of traditional frame editors. Note that in this context, the domain and range restrictions are *global*, and apply to every occurrence of the slot, whether explicit or implicit. A slot r can also be asserted to be transitive, functional or symmetric.

As well as standard class definitions (which are really a restricted form of subsumption/equivalence axiom), OilEd *axioms* can also be used to assert the *disjointness* or *equivalence* of classes (with the expected semantics) along with *coverings*. A covering asserts that every instance of the covered class must also be an instance of at least one of the covering classes. In addition, coverings can be said to be *disjoint*, in which case every instance of the covered class must be an instance of exactly one of the covering classes. Again, these axioms are not restricted to class names, but can involve arbitrary class expressions (anonymous frames or boolean combinations). This is a very powerful feature, exceeding the expressivity of traditional frame languages/editors, and is one of the main reasons for the high complexity of the underlying decision problem.

Limited functionality is provided to support the introduction and description of *individuals*—the intention within OilEd is that such individuals are for use within class descriptions, rather than supporting the production of large existential knowledge bases (it is supposed that RDF/RDFS will be used directly for this purpose). As an example, we may wish to define the class of Italians as being all those Persons who were born in Italy, where Italy is not a class but an individual.

As the FaCT system does not support reasoning with individuals, they are treated (for reasoning purposes) as disjoint primitive classes. This is not an ideal solution as it does lead to some inferences being lost, in particular those resulting from the interaction between individuals and cardinality constraints. E.g., it would not be possible to infer that Persons who are citizens of Italy, and of no other Country, are citizens of at most one Country. Work is currently underway to extend the FaCT reasoner to deal explicitly with such individuals, so that complete inference can be provided.

Concrete datatypes (string and integers), along with expressions concerning concrete datatypes (such as min, max or ranges) can also be used within class descriptions. However, the FaCT reasoner does not support reasoning over concrete datatypes, and at present OilEd simply ignores concrete datatype restrictions when reasoning about ontologies. The theory underyling concrete datatypes is, however, well understood [16], and work is also in progress to extend the FaCT reasoner with support for concrete datatypes.

3.1 Reasoning

In addition to the extended expressivity discussed above, OilEd's principal novelty is in its use of reasoning to check class consistency and infer subsumption relationships. Reasoning services are currently provided by the FaCT system, a DL classifier that offers sound and complete reasoning (satisfiability, subsumption and classification) for two DLs: $\mathcal{SHF}$ and $\mathcal{SHIQ}$. FaCT's most interesting features are its expressive logic (in particular the $\mathcal{SHIQ}$ reasoner), its optimised tableaux implementation (which has now become the standard for DL systems), and its CORBA based client-server architecture [17]. In principal any reasoner with the appropriate functionality/connectivity could be used.

The $\mathcal{SHIQ}$ language can completely capture OIL ontologies, with the exception of two recently added features: concrete datatypes (strings, numbers, etc.) and named individuals in class descriptions. As mentioned above, individuals can be dealt with by treating them as pairwise disjoint atomic classes (although with some loss of inferential power), while extending FaCT to deal with OIL's concrete datatypes should be relatively straightforward.

In the current version of OilEd, reasoning is performed on a "single-shot" basis, i.e., at some suitable point the user connects to the reasoner and requests verification of the ontology. Connection is via FaCT's CORBA based client-server interface, which has the advantage that FaCT servers(s) can be running either locally or remotely, and can provide a service to many OilEd users. Moreover, the FaCT system has reasoning engines for both $\mathcal{SHIQ}$ and $\mathcal{SHF}$ knowledge bases, and if both services are available the user can choose to connect to the faster $\mathcal{SHF}$ reasoner to verify an ontology that does not include either inverse slots or cardinality constraints. The current implementation simply informs the user if this is appropriate; future enhancements will include automatic selection of an appropriate reasoning service.

When verification is requested, the ontology is translated into an equivalent $\mathcal{SHIQ}$ (or $\mathcal{SHF}$) knowledge base and sent to the reasoner for classification. OilEd then queries the classified knowledge base, checking for inconsistent classes and implicit subsumption relationships. The results are reported to the user by highlighting inconsistent classes and rearranging the class hierarchy display to reflect any changes discovered. FaCT/OilEd does not provide any explanation of its inferences, although this would clearly be useful in ontology design [18]. Figure 3 shows the effects of classification on (part of) the hierarchy derived from the TAMBIS ontology. When verifying the ontology, a number of new subsump-

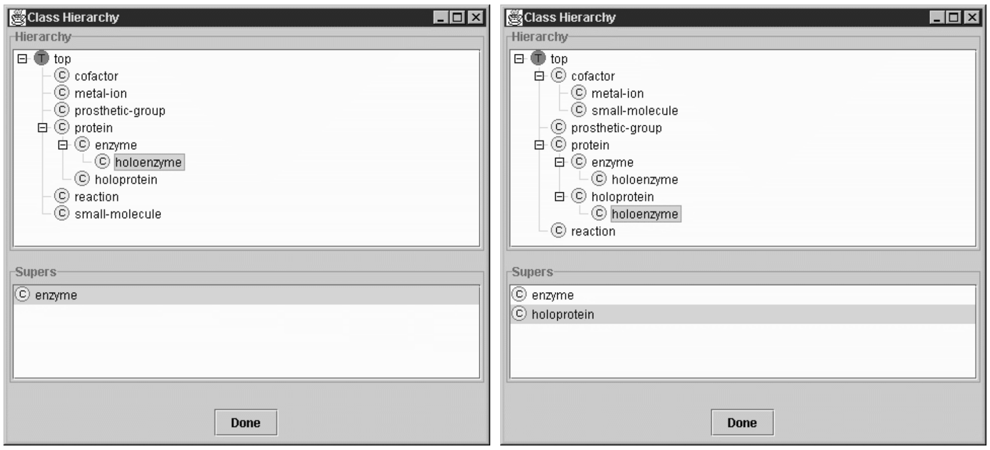

Figure 3: Hierarchy pre- and post-classification

tion relationships are discovered (due to the class definitions in the model). In particular we can see that, after verification, holoenzyme is not only an enzyme, but also a holoprotein, and that metal-ion and small-molecule are both subclasses of cofactor. During subsequent editing, changes to the ontology are not communicated to the reasoner instantaneously, but only when explicitly requested by the user. Future versions of OilEd may incorporate "real-time" reasoning support, but the simple interaction model described here was considered appropriate for the initial prototype.

4 Conclusion

We have presented OilEd, an ontology editor that has an easy to use frame interface, yet at the same time allows users to exploit the full power of an expressive web ontology language (OIL/DAML+OIL). We have also shown how OilEd uses reasoning to support ontology design and maintenance.

OilEd is a prototype, designed to test and demonstrate novel ideas, and it still lacks many features that would be required of a fully-fledged ontology development environment, e.g., it provides no support for versioning, or for working with multiple ontologies. Moreover, the reasoning support provided by the FaCT system is incomplete for OIL extended with concrete datatypes and individuals, and does not include additional services such as explanation. However, in spite of these shortcomings, OilEd is already sufficiently well developed to be a very useful tool, and to demonstrate the utility of OIL's integration of features from frame, DL and web languages.

References

[1] G. van Heijst, A. Schreiber, and B. Wielinga. Using explicit ontologies in KBS development. *Int. J. of Human-Computer Studies*, 46(2/3), 1997.

[2] D. L. McGuinness. Ontological issues for knowledge-enhanced search. In *Proc. of FOIS-98*, 1998.

[3] M. Uschold and M. Grüninger. Ontologies: Principles, Methods and Applications. *K. Eng. Review*, 11(2):93–136, 1996.

[4] T. Berners-Lee. *Weaving the Web.* Orion Business Books, 1999.

[5] D. Fensel *et al.* OIL in a nutshell. In *Proc. of EKAW-2000*, LNAI, 2000.

[6] S. Decker *et al.* The Semantic Web — on the respective roles of XML and RDF. *IEEE Internet Computing*, 2000.

[7] J. Hendler and D. L. McGuinness. The DARPA Agent Markup Language. *IEEE Intelligent Systems*, Jan 2001.

[8] S. Bechhofer, C.A. Goble, and I. Horrocks. DAML+OIL is not enough. In *To appear in SWWS-1, Semantic Web working symposium*, Stanford, CA, August 2001.

[9] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An Environment for Merging and Testing Large Ontologies. In *Proc. of KR-00*, 2000.

[10] J. Doyle and R. Patil. Two Theses of Knowledge Representation. *Artificial Intelligence*, 48:261–297, 1991.

[11] V. K. Chaudhri *et al.* OKBC: A programmatic foundation for knowledge base interoperability. In *Proc. of AAAI-98*, 1998.

[12] S. Staab and A. Maedche. Ontology engineering beyond the modeling of concepts and relations. In *Proc. of the ECAI'2000 Workshop on Application of Ontologies and Problem-Solving Methods*, 2000.

[13] W. E. Grosso *et al.* Knowledge Modeling at the Millennium (The Design and Evolution of Protégé-2000). In *Proc. of KAW99*, 1999.

[14] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Expressive Description Logics. In *Proc. of LPAR'99*, pages 161–180, 1999.

[15] I. Horrocks. Benchmark Analysis with FaCT. In *Proc. TABLEAUX 2000*, pages 62–66, 2000.

[16] F. Baader and P. Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In *Proc. of IJCAI-91*, pages 452–457, 1991.

[17] S. Bechhofer, I. Horrocks, P. F. Patel-Schneider, and S. Tessaris. A Proposal for a Description Logic Interface. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, pages 33–36, 1999.

[18] D. McGuinness and A. Borgida. Explaining Subsumption in Description Logics. In *Proc. of IJCAI-95*, pages 816–821, 1995.