Knowledge Representation on the Web

Stefan Decker[1], Dieter Fensel[2], Frank van Harmelen[2,3], Ian Horrocks[4], Sergey Melnik[1], Michel Klein[2] and Jeen Broekstra[3]

[1] AIFB, University of Karlsruhe, Germany

[2] Vrije Universiteit Amsterdam, Holland

[3] AIdministrator Nederland B.V.

[4] Department of Computer Science, University of Manchester, UK

Abstract

Exploiting the full potential of the World Wide Web will require semantic as well as syntactic interoperability. This can best be achieved by providing a further representation and inference layer that builds on existing and proposed web standards. The OIL language extends the RDF schema standard to provide just such a layer. It combines the most attractive features of frame based languages with the expressive power, formal rigour and reasoning services of a very expressive description logic.

1 Introduction

The World Wide Web has been made possible through a set of widely established standards which guarantee interoperability at various levels: the TCP/IP protocol has ensured that now nobody has to worry about transporting bits over the wire anymore. Similarly, HTTP and HTML have provided a standard way of retrieving and presenting hyperlinked text documents. Applications were able to use this common infrastructure and this has lead to the WWW as we know it now.

The current Web can be characterised as the second Web generation: the first generation Web started with handwritten HTML pages; the second generation made the step to machine generated and often active HTML pages. These generations of the Web were meant for direct human processing (reading, browsing, form-filling, etc). The third generation Web that we could call the "Knowledgeable Web", aims at machine processable meaning of information. This coincides with the vision that Tim Berners-Lee calls the Semantic Web in his recent book "Weaving the Web" [2]. The Knowledgeable Web will enable intelligent services such as information brokers, search agents, information filters etc.

The Semantic Web with machine processable information contents will only be possible when further levels of interoperability are established. Standards must be defined not only for the syntactic form of documents, but also for the form of their semantic contents. Such semantic interoperability is facilitated by recent W3C standardisation efforts, notably XML/XML Schema and RDF/RDF Schema.

In this paper, we make the following claims:

- A further representation and inference layer is needed on top of the currently available layers.
- This additional layer should be placed on top of the RDF layer, and not on top of the XML layer. This runs counter to many XML proponents, who recommend XML as the solution of the interoperability problems.

We will present the representation and inference language OIL[1] [6] and show how it can be embedded into the Semantic Web infrastructure by placing it on top of the RDF/RDF-Schema layer. OIL aims to combine the most attractive features of frame based languages (i.e., their more intuitive syntax and their acceptability within the ontology community) with the expressive power and formal rigour of a very expressive description logic. In fact the OIL language could be seen as nothing more that a frame-like syntax for the $\mathcal{SHIQ}$ DL. However, this is to underestimate the importance of the re-packaging and of the RDF/RDF-Schema embedding, the combination of which has already generated considerable interest in the web ontology community. Of course the embedding technique could be used with any DL, but the features of OIL/$\mathcal{SHIQ}$ (e.g., its support for role as well as concept hierarchies) make it particularly suitable as an extension for RDF/RDF-Schema.

2 Ontologies

Ontologies will play a crucial role in enabling the processing and sharing of knowledge between programs on the Web. Ontologies are generally defined as a "representation of a shared conceptualisation of a particular domain". They provide a shared and common understanding of a domain that can be communicated across people and application systems. They have been developed in Artificial Intelligence to facilitate knowledge sharing and reuse. Recent articles covering various aspects of ontologies can be found in [9, 10, 5].

An example of the use of ontologies on the Knowledgeable Web is in e-commerce sites where ontologies are needed (a) to enable machine-based communication between buyer and seller, (b) to enable vertical integration of markets (e.g. www.verticalnet.com), and (c) to leverage reusable descriptions between different marketplaces.

[1]Ontology Inference Layer.

A second example of the use of ontologies can be found in search engines. By using ontologies the search engines can escape from the current keyword-based approach, and can find pages that contain syntactically different, but semantically similar words (e.g. www.hotbot.com).

Typically, an ontology contains a hierarchical description of important concepts in a domain (is-a hierarchy), and describes crucial properties of each concept through an attribute-value mechanism. Additionally, further relations between concepts may be described through additional logical sentences. Finally, individuals in the domain of interest are assigned to one or more concepts in order to give them their proper type.

3 The OIL language

OIL is based on XOL [7] (which is an XML serialisation of the OKBC-lite knowledge model), with extensions that enable the full power of the $\mathcal{SHIQ}$ DL to be captured:

- Arbitrary boolean expressions (called *class expressions*) are allowed anywhere that a class name can appear.
- A slot definition can be treated as a class and can be used in class expressions.
- Class definitions have an (optional) additional field that specifies whether the class is primitive or non-primitive (the default is primitive).
- A class can be used as a slot `value` and is taken to specify that the slot must have at least one filler that is an instance of the given class.
- Global slot definitions are extended to allow the specification of superslots (subsuming slots) and of relation properties such as `transitive`, and `symmetrical`.
- The additional rules governing XOL documents (see [7]) are not required in OIL (e.g., there is no restriction on the ordering of class and slot definitions).

The extensions are designed to be backwards compatible with XOL so that, when no extensions are used, the result is an XOL/OKBC ontology.

OIL also restricts XOL in some respects.

- Initially, only conceptual modelling will be supported, i.e., individuals are not supported. This does not seem too onerous a restriction for an ontology specification language given that an ontology can be viewed as a kind of schema. Moreover, allowing individuals to occur in class definitions is equivalent to having extensionally defined classes, and this soon leads to very hard reasoning problems and even undecidability [8, 3, 1].

- The slot constraints `numeric-minimum` and `numeric-maximum` are not supported. Future extensions of OIL may support concrete data types (including numbers and numeric ranges).

- Collection types other than `set` are not supported.

- Slot `inverse` can only be specified in global slot definitions: naming the inverse of a relation only seems to make sense when applied globally.

The semantics of OIL rely on a translation into the $\mathcal{SHIQ}$ description logic. $\mathcal{SHIQ}$ has a highly expressive concept language that is able to fully capture the OIL core language, and we will define a satisfiability preserving translation $\sigma(\cdot)$ that maps OIL ontologies into $\mathcal{SHIQ}$ terminologies. This has the added benefit that an existing $\mathcal{SHIQ}$ reasoner implemented in the FaCT system can be used to reason with OIL ontologies.

An OIL ontology $\mathcal{O}$ consists of a list $d_1, \ldots, d_n$, where each d_i is either a class definition or a slot definition. This list of definitions is translated into a $\mathcal{SHIQ}$ terminology $\mathcal{T}$ (a set of axioms) as follows:

$$\sigma(d_1, \ldots, d_n) = \bigcup_{i=1,\ldots,n} \sigma(d_i)$$

A class definition is either a pair $\langle \mathsf{CN}, D\rangle$ or a triple $\langle \mathsf{CN}, P, D\rangle$, where CN is a class name, D is a class description and P is either `primitive` or `defined`; $\langle \mathsf{CN}, D\rangle$ is equivalent to $\langle \mathsf{CN}, \texttt{primitive}, D\rangle$. A class definition $\langle \mathsf{CN}, \texttt{primitive}, D\rangle$ is written $\mathsf{CN} \sqsubseteq D$ (it states that CN is a subclass of the class described by D) and a class definition $\langle \mathsf{CN}, \texttt{defined}, D\rangle$ is written $\mathsf{CN} \doteq D$ (it states that CN is equivalent to the class described by D).

A class description D consists of an optional `subclass-of` component, itself a list of one or more `class-expressions` $C_1, \ldots, C_n$, followed by a list of zero or more `slot-constraints` $A_1, \ldots, A_m$. We will write such a class description as

$$[C_1, \ldots, C_n, A_1, \ldots, A_m].$$

A `class-expression` is either a class name CN, a `slot-constraint`, a conjunction of class expressions, written $C_1 \sqcap \ldots \sqcap C_n$, a disjunction of class expressions, written $C_1 \sqcup \ldots \sqcup C_n$ or a negated class expression, written $\neg C$. A `slot-constraint` consists of a slot name SN followed by one or more constraints that apply to the slot, written $\mathsf{SN}[a_1, \ldots, a_n]$. Each constraint can be either:

- A `value` constraint with a list of one or more class-expressions, written $\exists C_1, \ldots, C_n$.

- A `value-type` constraint with a list of one or more class-expressions, written $\forall C_1, \ldots, C_n$.

$$
\begin{array}{rcl}
\sigma(\mathsf{CN} \sqsubseteq D) & = & \{\sigma(\mathsf{CN}) \sqsubseteq \sigma(D)\} \\
\sigma(\mathsf{CN} \doteq D) & = & \{\sigma(\mathsf{CN}) \sqsubseteq \sigma(D), \sigma(D) \sqsubseteq \sigma(\mathsf{CN})\} \\
\sigma([C_1, \ldots, C_n, A_1, \ldots, A_m]) & = & \top \sqcap \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n) \sqcap \sigma(A_1) \sqcap \ldots \sqcap \sigma(A_m) \\
\sigma(\mathsf{CN}) & = & \mathsf{CN} \\
\sigma(\top) & = & \top \\
\sigma(C_1 \sqcap \ldots \sqcap C_n) & = & \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n) \\
\sigma(C_1 \sqcup \ldots \sqcup C_n) & = & \sigma(C_1) \sqcup \ldots \sqcup \sigma(C_n) \\
\sigma(\neg C) & = & \neg\sigma(C) \\
\sigma(\textsf{\textit{SN}}[a_1, \ldots, a_n]) & = & \sigma(\textsf{\textit{SN}}(a_1)) \sqcap \ldots \sqcap \sigma(\textsf{\textit{SN}}(a_n)) \\
\sigma(\textsf{\textit{SN}}(\exists C_1, \ldots, C_n)) & = & \exists \textsf{\textit{SN}}.\sigma(C_1) \sqcap \ldots \sqcap \exists \textsf{\textit{SN}}.\sigma(C_n) \\
\sigma(\textsf{\textit{SN}}(\forall C_1, \ldots, C_n)) & = & \forall \textsf{\textit{SN}}.\sigma(C_1) \sqcap \ldots \sqcap \forall \textsf{\textit{SN}}.\sigma(C_n) \\
\sigma(\textsf{\textit{SN}}(\leqslant n, C)) & = & \leqslant n \textsf{\textit{SN}}.\sigma(C) \\
\sigma(\textsf{\textit{SN}}(\geqslant n, C)) & = & \geqslant n \textsf{\textit{SN}}.\sigma(C) \\
\sigma(\textsf{\textit{SN}}(= n, C)) & = & \leqslant n \textsf{\textit{SN}}.\sigma(C) \sqcap \geqslant n \textsf{\textit{SN}}.\sigma(C)
\end{array}
$$

Figure 1: Translation of OIL class definitions into $\mathcal{SHIQ}$

- A `max-cardinality` constraint with a non-negative integer n followed (optionally) by a class expression C, written $\leqslant n, C$ ($\leqslant n, \top$ if the class expression is omitted).

- A `min-cardinality` constraint with a non-negative integer n followed (optionally) by a class expression C, written $\geqslant n, C$ ($\geqslant n, \top$ if the class expression is omitted).

- A `cardinality` constraint with a non-negative integer n followed (optionally) by a class expression C, written $=n, C$ ($=n, \top$ if the class expression is omitted).

In order to maintain the decidability of the language, cardinality constraints can only be applied to *simple* slots. A simple slot is one that is neither transitive nor has any transitive subslots. However, as the transitivity of a slot can be inferred (e.g., from the fact that the inverse of the slot is a transitive slot), simple slot is defined in terms of the translation into $\mathcal{SHIQ}$: a slot *SN* in an ontology $\mathcal{O}$ is a simple slot iff $\sigma(\textsf{\textit{SN}})$ is a simple role in the $\mathcal{SHIQ}$ terminology $\sigma(\mathcal{O})$.

We can now define how the function $\sigma(\cdot)$ maps an OIL class definition into a set of $\mathcal{SHIQ}$ axioms. The definition is given in Figure 1, where CN is a class name (or a $\mathcal{SHIQ}$ concept name), *SN* is a slot name (or $\mathcal{SHIQ}$ role name), D is a class description, C (possibly subscripted) is a class expression, A (possibly subscripted) is a slot constraint, a_i is a constraint (on a slot) and n is a non-negative integer.

A slot definition is a pair $\langle \textsf{\textit{SN}}, D\rangle$, where *SN* is a slot name and D is a slot description. A slot description D consists of an optional `subslot-of` component,

$$
\begin{array}{rcl}
\sigma(\mathit{SN}[\mathit{RN}_1,\ldots,\mathit{RN}_n,S_1,\ldots,S_m]) & = & \sigma(\mathit{SN}[\mathit{RN}_1,\ldots,\mathit{RN}_n] \cup \sigma(\mathit{SN}[S_1,\ldots,S_m]) \\
\sigma(\mathit{SN}[\mathit{RN}_1,\ldots,\mathit{RN}_n] & = & \bigcup_{i=1,\ldots,n} \sigma(\mathit{SN} \sqsubseteq \mathit{RN}_i) \\
\sigma(\mathit{SN}[S_1,\ldots,S_m]) & = & \bigcup_{i=1,\ldots,m} \sigma(\mathit{SN}(S_i)) \\
\sigma(\mathit{SN} \sqsubseteq \mathit{RN}) & = & \{\mathit{SN} \sqsubseteq \mathit{RN}\} \\
\sigma(\mathit{SN}(\downarrow [C_1,\ldots,C_n])) & = & \bigcup_{i=1,\ldots,n} \{\exists \mathit{SN}.\top \sqsubseteq \sigma(C_i)\} \\
\sigma(\mathit{SN}(\uparrow [C_1,\ldots,C_n])) & = & \bigcup_{i=1,\ldots,n} \{\top \sqsubseteq \forall \mathit{SN}.\sigma(C_i)\} \\
\sigma(\mathit{SN}(^-\mathit{RN})) & = & \{\mathit{SN}^- \sqsubseteq \mathit{RN}, \mathit{RN} \sqsubseteq \mathit{SN}^-\} \\
\sigma(\mathit{SN}([P_1,\ldots,P_n])) & = & \bigcup_{i=1,\ldots,n} \{\sigma(\mathit{SN}(P_i))\} \\
\sigma(\mathit{SN}(+)) & = & \{\mathit{SN} \in \mathrm{S}_+\} \\
\sigma(\mathit{SN}(\leftrightarrow)) & = & \{\mathit{SN}^- \sqsubseteq \mathit{SN}, \mathit{SN} \sqsubseteq \mathit{SN}^-\}
\end{array}
$$

Figure 2: Translation of OIL slot definitions into $\mathcal{SHIQ}$

itself a list of one or more slot names $\mathit{RN}_1,\ldots,\mathit{RN}_n$, followed by a list of zero or more global slot constraints (e.g., `inverse`) $S_1,\ldots,S_m$. We will write such a slot definition as:

$$\mathit{SN}[\mathit{RN}_1,\ldots,\mathit{RN}_n,S_1,\ldots,S_m]$$

Each global constraint S_i on SN can be either:

- A `domain` constraint with a list of one or more class-expressions, written $\downarrow [C_1,\ldots,C_n]$.
- A `range` constraint with a list of one or more class-expressions, written $\uparrow [C_1,\ldots,C_n]$.
- An `inverse` constraint with a slot name RN, written $^-\mathit{RN}$.
- A `properties` constraint with a list of one or more properties, written $[P_1,\ldots,P_n]$. Valid properties are `transitive`, written $+$ and `symmetrical`, written $\leftrightarrow$.

We can now define how the function $\sigma(\cdot)$ maps an OIL slot definition into a set of $\mathcal{SHIQ}$ axioms. The definition is given in Figure 2, where RN and SN are slot names (or $\mathcal{SHIQ}$ role names), C_i is a class expression, S_i is a global slot constraint and P_i is a property.

4 Using OIL to Enrich RDF/RDF Schema

One of the key ideas behind OIL is that its much more expressive modelling primitives can be used to enrich RDF/RDF Schema. We have chosen RDF rather than XML because XML is just a formalism for defining a grammar: it can be used to represent the structure of a document, but does not impose any common interpretation no the data contained in the document.

RDF is designed to facilitate semantic interoperability by representing a domain model in terms of simple object-relation-object triples, and techniques from Knowledge Representation can be used to help find mappings between two RDF descriptions. Of course this does not solve the general interoperability problem (i.e., finding semantic-preserving mappings between objects), but the usage of RDF for data interchange raises the level of potential reuse much beyond the parser reuse which is all that one would obtain from using plain XML. Moreover, since RDF describes a layer independent of XML, an RDF domain model (and software using the RDF model) could still be used, even if XML syntax changes or becomes obsolescent.

Of course we would ideally like a universally shared KR language to support the Semantic Web. For a variety of pragmatic and technological reasons, this ideal is not achievable in practice, and we will have to live with a multitude of meta-data representations. RDF is said to contain as much KR technology as can be shared between widely varying meta-data languages–which is not very much! However, the RDF Schema language is powerful enough to define richer languages on top of the relatively limited primitives of RDF.

Defining an ontology in RDF means defining an RDF Schema which in turn defines all the terms and relationships of the particular language. In this ontology the subclass-of primitive is a relation. This identifies the two arguments of "subclass-of" as objects: a class (the subclass) and a class-expression (the superclass). The view of the class-expression as a object is justified by the fact that the expression defines a new (unnamed) class. The remaining OIL modelling primitives are similarly defined in the schema (see [6] for full details).

Since every ontology (RDF Schema) uses its own namespace [4], terms from different ontologies can be mixed in one RDF document without confusion. Since RDF defines a clear object structure, it is possible to make assertions with one language about an object defined in terms of another language. Note that this is not possible in XML: since there is no defined meaning for a particular tag (Object, Attribute, Value etc.), nothing can be assumed about the object structure.

Our proposal can be summarised as follows:

1. Use RDF Schema to describe the OIL modelling primitives.

2. Use the resulting RDF Schema document (containing the meta-ontology of OIL) to describe a specific ontology in OIL.

3. Use the RDF Schema documents from 1 and 2 to describe instances of the specific OIL ontology modelled in 2.

Although the same technique could be applied to any KR language, the fact that OIL supports both of the basic RDF schema modelling primitives (rdfs:subClassOf and rdfs:subPropertyOf) makes it particularly well suited to this purpose, as significant parts of OIL ontologies will be available to any RDF schema aware application. Moreover, the direct correspondence with the $\mathcal{SHIQ}$ DL can be exploited to provide reasoning services for OIL. As well as supporting ontology design, these reasoning services could be used to make inferred subsumption relations available to any RDF schema aware application by adding explicit rdfs:subClassOf statements to the ontology.

Initially, OIL's main use may be in ontology design, with the resulting ontologies being used by applications that are only RDF schema aware, and simply ignore any additional knowledge captured by OIL. However, in the longer run we hope to see the development of "OIL aware" applications that will enhance the services they provide by exploiting the richer semantic content of OIL ontologies.

5 Discussion

In this paper we have argued that semantic interoperability will be a *sine qua non* for the Semantic Web, and that such semantic interoperability must be achieved by exploiting the current RDF proposals. The RDF datamodel makes approaches from AI and Knowledge Engineering to establish semantic interoperability directly applicable. Furthermore, it is highly extensible. We have demonstrated this by applying it to a particular ontological modelling language, OIL. A similar strategy should apply to any knowledge modelling language, although we have argued that OIL is particularly suitable for extending RDF.

The arguments in this paper are an important message to at least two different communities.

The Web community is currently regarding XML as the most important step towards semantic integration. We have argued why this cannot be true in the long run, and why RDF is a much better platform for this.

The AI community is currently very much interested in applying many of its techniques to the Web. We have shown a generic method that can be used to Web-enable arbitrary Knowledge Representation languages. This is an important step towards the realization of the dream of the Semantic Web.

References

[1] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In *Proc. of CSL'99*, number 1683, pages 307–321, 1999.

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

[3] P. Blackburn and J. Seligman. What are hybrid languages? In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, editors, *Advances in Modal Logic*, volume 1, pages 41–62. CSLI Publications, Stanford University, 1998.

[4] T. Bray, D. Hollander, and Andrew Layman (eds.). Namespaces in xml. World Wide Web Consortium Recommendation, Jan 1999. `http://www.w3.org/TR/REC-xml-names/`.

[5] A. Gomez Perez and V. R. Benjamins. Applications of ontologies and problem-solving methods. *AI-Magazine*, 20(1):119–122, 1999.

[6] I. Horrocks, D. Fensel, C. Goble, F. Van Harmelen, J. Broekstra, M. Klein, and S. Staab. The ontology inference layer oil. Technical report, Free University of Amsterdam, 2000. `http://www.ontoknowledge.org/oil/`.

[7] P. D. Karp, V. K. Chaudhri, and J. Thomere. XOL: An XML-based ontology exchange language. Version 0.3, July 1999.

[8] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.

[9] M. Uschold and M. Grüninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2), 1996.

[10] G. van Heijst, A. Th. Schreiber, and B. J. Wielinga. Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies*, 46(2/3):183–292, 1997. `http://ksi.cpsc.ucalgary.ca/IJHCS/VH/`.