FAQs on OIL: the Ontology Inference Layer

Frank van Harmelen
(Vrije Universiteit Amsterdam)

Ian Horrocks
(University of Manchester)

What is OIL trying to achieve

The current Web is entirely aimed at human readers. Machines are oblivious to the actual information content: web browsers, web servers and even search-engines do not really distinguish weather-forecasts from scientific papers, and cannot tell a personal home-page from a major corporate website. This inability to process the contents of information by machines seriously hampers the functionality of the current Web. Computers are limited to transmit and present information on the Web, and cannot really help us in processing this information. The vision of the Semantic Web aims at creating a Web where information can be "understood" by machines as well as by humans. This of course requires that information is represented in such a way that its meaning (its "semantics") is in a machine-accessible form. OIL is designed to be exactly such a representation of machine-accessible semantics of information on the Web.

How is OIL trying to achieve this

OIL synthesizes work from three different communities to achieve the ambitious aim of providing a general purpose markup-language for the Semantic Web:

- **Frame-based systems**: frame-based languages have a long history in AI. Their central modelling primitives are classes (known as frames) with properties (known as slots). A frame provide a context for modelling a class, which is generally defined a subclass of one or more other classes, with slot-value pairs being used to specify additional constraints on instances of the new class. Many frame-based systems and languages with many additional refinements of these modelling primitives have been developed and, renamed as object-orientation they have been very successful in the software engineering community. OIL incorporates the essential modelling primitives of frame-based systems, being based on the notion of a concept and the definition of its superclasses and slots. OIL also treats slots as first class objects that can have their own properties (e.g., domain and range) and can be arranged in a hierarchy.

- **Description logics**: Description logics (DLs) have been developed in knowledge-representation research, and describe knowledge in terms of concepts (comparable to classes, or frames) and roles (comparable to slots in frame systems). An important aspect of DLs is that they have very well understood theoretical properties, and that the meaning of any expression in a DL can be described in a mathematically precise way; this enables reasoning with concept descriptions and the automatic derivation of classification taxonomies. There are now efficient implementations
of DL reasoners able to perform these tasks. OIL inherits from DLs both their formal semantics and efficient reasoning support.

- **Web standards: XML and RDF.** Besides modelling primitives (provided by frame systems) and their semantics (provided by description logics), we have to decide about the syntax of a markup language for the Semantic Web. Any such syntax must be formulated using existing W3C standards for information representation. First, OIL has a well-defined syntax in XML based on a DTD and a XML schema definition. Second, OIL is defined as an extension of the Resource Description Framework RDF and its schema definition language RDF Schema. RDF Schema provides two important contributions: a standard set of modelling primitives like instance-of and subclass-of relationships, and a standardized syntax for writing such writing class-hierarchies. OIL extends this approach to a full-blown modelling language.

**What does OIL look like**

Below, we give a very simple example of an OIL ontology. It only illustrates the most basic constructs of OIL.

```xml
<class-def Product>
    <slot-def Price>
        <domain Product/>
    </slot-def>
    <slot-def ManufacturedBy>
        <domain Product/>
    </slot-def>
</class-def>

<class-def PrintingAndDigitalImagingProduct>
    <subclass-of Product/>
</class-def>

<class-def HPPrintProduct>
    <subclass-of Product/>
    <slot-constraint ManufacturedBy>
        <has-value "Hewlett Packard"/>
    </slot-constraint>
</class-def>

<class-def Printer>
    <subclass-of PrintingAndDigitalImagingProduct/>
    <slot-def PrinterTechnology>
        <domain Printer/>
    </slot-def>
    <slot-def PrintingSpeed>
        <domain Printer/>
    </slot-def>
    <slot-def PrintingResolution>
        <domain Printer/>
    </slot-def>
</class-def>

<class-def PrinterForPersonalUse>
    <subclass-of Printer/>
</class-def>

<class-def LaserJetPrinter>
    <subclass-of Printer/>
    <slot-constraint PrintingTechnology>
        <has-value "Laser Jet"/>
    </slot-constraint>
</class-def>

<class-def HPLaserJetPrinter>
    <subclass-of LaserJetPrinter and HPProduct/>
</class-def>

<class-def HPLaserJet1100Series>
    <subclass-of HPLaserJetPrinter and PrinterForPersonalUse/>
    <slot-constraint PrintingSpeed>
        <has-value "8 ppm"/>
    </slot-constraint>
    <slot-constraint PrintingResolution>
        <has-value "600 dpi"/>
    </slot-constraint>
</class-def>

<class-def HPLaserJet1100se>
    <subclass-of HPLaserJet1100Series/>
    <slot-constraint Price>
        <has-value "$479"/>
    </slot-constraint>
</class-def>

<class-def HPLaserJet1100xi>
    <subclass-of HPLaserJet1100Series/>
    <slot-constraint Price>
        <has-value "$399"/>
    </slot-constraint>
</class-def>
```

This defines a number of classes and organises them in a class-hierarchy (e.g. HPProduct is a subclass
of `Product`). Various properties ("slots") are defined, together with the classes to which they apply (e.g. a `Price` is a property of any `Product`, but a `PrintingResolution` can only be stated for a `Printer` (an indirect subclass of `Product`). For certain classes, these properties have restricted values (e.g. the `Price` of any `HPLaserJet1100se` is restricted to be $479). In OIL, classes can also be combined using logical expressions, for example: an `HPPrinter` is both an `HPProduct` and a `Printer` (and consequently inherits the properties from both these classes).

**What does the acronym "OIL" mean?**

There are a number of possible meanings of the acronym: "Ontology Inference Layer", or "Ontology Interchange Language", but all of the contain the word "Ontology". An ontology is a consensual, shared and formal description of the concepts that are important in a given domain. Typically, an ontology identifies classes of objects that are important in a domain, and organises these classes in a subclass-hierarchy. Each class is characterised by properties shared by all elements in that class. Important relations between classes or between the elements of these classes are also part of an ontology. Ontologies are now an important notion in such diverse areas as knowledge representation, natural language processing, information retrieval, databases, knowledge management, multi-agent systems, and others. They are widely considered to be a crucial ingredient for the infrastructure of the Semantic Web.

**Which applications will be enabled by OIL?**

Machine-processable representations of ontologies will be crucial to many applications of the semantic Web. We briefly mention only a few:

- **search engines**: current search engines are seriously limited by their reliance on keyword-matching. They are unable to find relevant information that is described in different terms, they often return information that uses the same words with a different meaning, and they are unable to combine information from multiple sources. These problems can be alleviated if search engines no longer search for matching keywords, but search on the semantic concepts that underly the information in web-pages.

- **E-commerce**: currently, consumers can only compare on-line shops by visiting each shop themselves and doing the comparison. So called shopbots that try to perform this task do this by so-called "screen-scraping": retrieving the information by interpreting regularities in the lay-out of the web-pages of the various shops. They typically only retrieve limited information from the various shops (e.g. price), and ignore information such as shipping conditions which are harder to retrieve. In addition, they are cumbersome to construct, and hard to maintain (they must be updated every time a web-shop changes the layout of its pages). Comparison-shopping will become only really possible when web-shops offer their catalogues in machine-processable formats, with links to explicit and shared ontologies that can be used to construct mappings between these catalogues.

- **knowledge management**: an increasing number of companies is relying on intra-net technology as a knowledge-repository for their employees. Traditional document-management systems provide insufficient means to structure and access the knowledge in such a repository. Explicit ontologies are the most promising technical vehicle for transforming document repositories into proper knowledge repositories.

**What are the design principles behind OIL**

The following have been important reasons motivating the design of OIL:

- maximising compatibility with existing W3C standards, as XML and RDF;
- maximising partial interpretability by less semantically aware processors;
providing modelling primitives that have proven useful for large user communities;
maximising expressiveness to enable modelling a wide variety of ontologies;
providing a formal semantics (a mathematically precise description of the meaning of every expression) in order to facilitate machine interpretation of that semantics;
enabling sound, complete and efficient reasoning services, if necessary by limiting the expressiveness of the language.

Which OIL tools are currently available

Ontology editors help human knowledge engineers to develop and maintain ontologies. They support the definition and modification of concepts, slots, axioms and constraints, as well as enabling the inspection, browsing and codifying of the resulting ontologies. Currently, two editors for OIL are available and a third one is under development:

- OntoEdit (Ontology Engineering Environment, http://ontoserver.aifb.uni-karlsruhe.de/ontoedit) developed at the Knowledge Management Group of the AIFB Institute at the University of Karlsruhe,
- OILedit, a freely available and customized editor for OIL, developed by the University of Manchester.
- Prote'ge' (http://www.smi.stanford.edu/projects/protege/) an ontology editor built at the University of Stanford. Currently it only supports RDF, but work is starting on extending Prote'ge' to OIL.

Inference engines can be used to reason about ontologies, helping both to build them and to use them for advanced information access and navigation. OIL uses the FaCT system (Fast Classification of Terminologies, http://www.cs.man.ac.uk/fact) to provide reasoning support for ontology design, integration and verification. FaCT is heavily optimised to deal with very large ontologies. It can check the consistency of thousands of classes and automatically derive their underlying class-hierarchy in a matter of seconds running on standard desk-top hardware.

How does OIL relate to RDF/RDF Schema

The above example was stated in OIL's presentation syntax, which is intended for human readers and writers of OIL ontologies. For machines, OIL uses RDF as its syntax. OIL exploits as much as possible the modelling primitives of RDF Schema. This provides crucial backwards compatibility, allowing OIL ontologies to be treated as extensions of RDF and RDF Schema ontologies, and making OIL ontologies available not only to OIL-aware applications, but also to applications that are only RDF-aware: such RDF-aware applications can still process and reason with significant portions of OIL-ontologies. For illustration purposes, the last class of the above example in RDF syntax would look like:

```xml
<rdfs:Class rdf:ID="HPLaserJet1100xi">
  <rdfs:subClassOf rdf:resource="#HPLaserJet1100Series"/>
  <oil:hasPropertyRestriction>
    <oil:HasValue>
      <oil:onProperty rdf:resource="#Price"/>
      <oil:toConcreteType>399</oil:toConcreteType>
    </oil:HasValue>
  </oil:hasPropertyRestriction>
</rdfs:Class>
```

To a program that is only RDF-aware (and not OIL-aware), this would still be interpretable as saying that the 1100xi printers are a special type of the 1100 Series printers. The specific restriction that the 1100xi costs $399 would only be available to OIL-aware programs.

How is OIL different from DAML
The DAML language inherits many aspects from OIL, and the capabilities of the two languages are relatively similar:

- both support hierarchies of classes and properties based on sub-class and sub-property relations.
- both allow classes to be built from other classes using arbitrary combinations of intersection (AND), union (OR) and complement (NOT);
- both allow the domain, range and cardinality of properties to be restricted;
- both support transitive and inverse properties;
- both support concrete data types (integers, strings, etc.); However, there are also some important differences, which we can only briefly discuss here:
- OIL achieves a greater backward compatibility with RDF Schema than DAML.
- OIL has been designed to enable reasoning services that are sound and complete as well as efficient. Some constructions in DAML make similar reasoning services for DAML impossible.
- OIL one can state either sufficient conditions for a class, or conditions that are both sufficient and necessary. This last option makes it possible to perform automatic classification: given a specific object in a domain, automatically decide to which classes this object belongs. In DAML this distinction is not as well developed as in OIL.
- DAML allows the specification of default values: values that can be assumed for a given property when no other value is specified. OIL avoids such default values, because no clear formal semantics for default values exists.

Who is funding and doing the work on OIL?

The OIL initiative is funded by the European Union IST programme for Information Society Technologies under the On-To-Knowledge project (IST-1999-1013) and IBROW (IST-1999-19005). Work is carried out by the participants in this project, and a large number of parties, both academic, commercial and institutional, outside this consortium interested in furthering OIL's development.

Where can I find out more about OIL

OIL's homepage is at [http://www.ontoknowledge.org/oil](http://www.ontoknowledge.org/oil). This page gives access to definitions of the syntax of OIL, papers and presentations explaining OIL (ranging from the very introductory to the very formal), case-studies using OIL, and tools that have been developed for OIL.