THE BRITISH COMPUTER SOCIETY

Report of the Roger Needham Lecture – 7 December 2005

*An account prepared in collaboration with Ian Horrocks
by Conrad Taylor of the BCS Electronic Publishing Specialist Group*

Ontologies and the Semantic Web

A lecture by Professor Ian Horrocks

Introduction

The Roger Needham Award, which is sponsored by Microsoft Research, is an annual prize established by the BCS in memory of the late Professor Roger Needham[1]. It is awarded for a distinguished research contribution in computer science by a UK-based researcher, within ten years of receiving his or her PhD.

The Roger Needham Award consists of a financial prize, plus the opportunity to present a lecture. The Roger Needham Lecture for 2005 was delivered at the Royal Society on 7 December 2005 by Professor Ian Horrocks, a member of the Information Management Group at the School of Computer Science of the University of Manchester.

"Today's Web is good, but it's not good enough"

Professor Horrocks started his lecture by introducing the thinking behind the Semantic Web. The idea of the Semantic Web has arisen from a critique of the existing Web.

Obviously the Web has been tremendously successful, is used by an unprecedented number of people for a computer system, and is extremely useful. However, it can also be very frustrating to use, and the reason is because it is a very simple artefact, scarcely more than distributed hypermedia. Finding useful information on the Web is quite a hit-and-miss process – a mix of keyword-based searching, plus some browsing around.

The information on the World Wide Web has been designed for human consumption. The annotations that website creators add to information – the mark-up – is intended to tell our browsers how to present it on the screen. As humans, we are able to make sense of the structure and significance of the information with reference to how it is laid out.

This relatively simple model for the Web wasn't what Tim Berners-Lee, the English scientist who was central to developing the Web, had in mind for it – or so he now tells us, at any rate. Tim says that he always intended the Web to be a set of connected applications forming a consistent logical web of data. The idea was that data would have well-defined meaning, and this would enable computers and people to work on the data together, in co-operation. This vision of what the Web perhaps always should have been, and where we hope it's going in the future, is what has become known as the **Semantic Web**.

1. Roger Needham, 1935–2003, was Professor of Computing Systems at the University of Cambridge and also head of its Computer Laboraroy. In 1997 he left the University and became the first director of Microsoft Research in Cambridge.

How this report was prepared: An audio recording was made of Ian Horrocks' lecture; this and the accompanying Powerpoint slides were made available to Conrad Taylor of BCS-EPSG, who wrote up this account, supplementing it with notes, references and background explanations. These were then checked by Ian Horrocks and amended accordingly. Conrad Taylor also redrew some diagrams, and replaced the original cartoons used in the lecture slides to avoid problems of copyright.

Problems with today's 'Syntactic Web'

Why might we want a Semantic Web? By way of illustration, Ian gave an example of the kind of thing that can go wrong using today's Web (we might call it the 'Syntactic Web'). On one occasion when preparing slides for a talk, Ian had wanted to acknowledge some people he had worked with: Peter Patel-Schneider, Frank van Harmelin and Alan Rector. He went looking for images of these three men using Google Image Search. Simply by typing *Peter Patel-Schneider* into Google, Ian found an image of him, and a picture of Frank was just as easy to track down. But when he typed in *Alan Rector* he got an image of a priest in vestments, a total stranger – not what he wanted at all. It is quite easy to figure out why this happened. Google had found a Web page featuring the Reverend ***Alan*** M Gates, an associate ***rector*** at the Church of the Holy Spirit, Lake Forest, Illinois. Which is the sort of less than useful result that keyword searching against unstructured text often returns.[2]

We can imagine more complex tasks that turn out to be very difficult or even impossible using the existing Web. A Manchester colleague of Ian's pointed out one such example. She had been searching for information about animals that use echo-location, but that are neither bats or dolphins. (Barn-owls, for example, use echo-location for night hunting.) But if you try even quite a sophisticated keyword-search – something like "animal sonar NOT bats NOT dolphins" – you are likely to eliminate almost every page that would be relevant to your search task, because such pages will usually mention either bats or dolphins, as well as these other animals you are interested in.

As for everyday use of the Web, it's nice that we can now book a holiday while sitting at our computers, but it can also be frustrating. You move from one site to another, comparing prices or itineraries… the sites all ask you to enter the same information – the day you're travelling, the day you're coming back… It would be really nice to automate this kind of process, but with the existing Web it is difficult, because the sites that provide these services are designed for human beings, not for machines to use.

Computers are increasingly used for performing scientific experiments; a practice that's being called 'e-science'. For example, a biologist may have a DNA sequence, and might want to identify genes in the sequence, determine which proteins would be produced by those genes, and then figure out which biological processes these genes control. Now, there are resources on the Web that can perform these functions – databases and other tools – but it's difficult to put them together to perform this kind of experiment, certainly in any automated way. Biologists and other scientists are therefore spending a lot of time devising Perl scripts to bolt these systems together. It would be nice if we could do it more automatically, and find these kinds of services more automatically, said Ian.

Some articles about the Semantic Web, such as the famous article in *Scientific American* by Tim Berners-Lee, James Hendler and Ora Lassila,[3] express some quite ambitious goals for what we might want from it: for example, to be able to delegate complex tasks to software 'agents', such as to book a holiday somewhere warm, not too far away, and where they speak either French or English. "I'm not sure I really believe in this happening in the very

2. To be fair to Google, they have made certain policy changes since Ian performed this search. Now, if you type in *Alan Rector*, those pages will be presented first where the words appear closest together, and Professor Rector's page is top of the list.
3. *The Semantic Web* by Tim Berners-Lee, James Hendler and Ora Lassila. Scientific American, May 2001.

near future," said Ian. "I hardly trust *myself* to book a holiday – I'm not sure I'd trust a computer program to do it for me! But we can think of these goals as a grand challenge for what we might like to do with the Web in the future – a sort of robo-football of the Semantic Web."

Extending the job that mark-up does

Why is it so difficult to carry out these kinds of automated tasks with the World Wide Web? Presenting us with a BCS Web page as an example, Ian pointed out that the function of the HTML mark-up is to do such things as make some of the text stand out visually by being bold and blue, and to make other parts of the page function as hypertext links to other resources. As for the semantic content of the page – what it actually means – it is easy for us humans to figure it out, but very difficult for a machine to understand. One would need to equip machines with natural language understanding at the very least. And some of the information is buried inside images: even more of a challenge.

The proposed solution is to add **explicit semantic annotations** to Web resources: that is, annotations which say something about what the information means. Going back to his 'Alan Rector' versus 'Alan the Rector' example, Ian showed how additional mark-up might improve the chances of being able to process the text more accurately:

```
Dr. <Person>Alan Rector</Person>, <Job>Professor of Computer Science</Job>,
University of Manchester
Rev. <Person>Alan M. Gates</Person>, <Job>Associate Rector</Job> of the
Church of the Holy Spirit, Lake Forest. Illinois.
```

Giving semantics to annotations

Adding annotative mark-up like this does make the purpose of information a bit easier to understand, a bit more machine-handleable, but Ian thought that by now we might be asking "But – where do the semantics come in?" After all, so far it looks as if all we have done is to push the problem one level back, from a problem of understanding natural language to a problem of understanding the language in the labels. "Painting labels on everything isn't immediately going to solve the problem," said Ian.

One of the most obvious solutions is to try to establish external agreements about the meaning of a suitable set of labels. An good example of this approach is the Dublin Core Metadata Initiative. This approach depends upon standardisation activity: people go off into committees, decide that they want a set of tags such as <Author> and <Subject> and <DatePublished> and so on, and the community then uses those standardised tags.

What's wrong with this? Well, it provides quite limited flexibility and extensibility, and it puts a rigid limit on the number of things that can be expressed by the metadata scheme. It's also rather a cumbersome process, because every time you want to annotate some new domain, you have to a set up a committee and wait several years until they come back with a standardised list of tags to use.

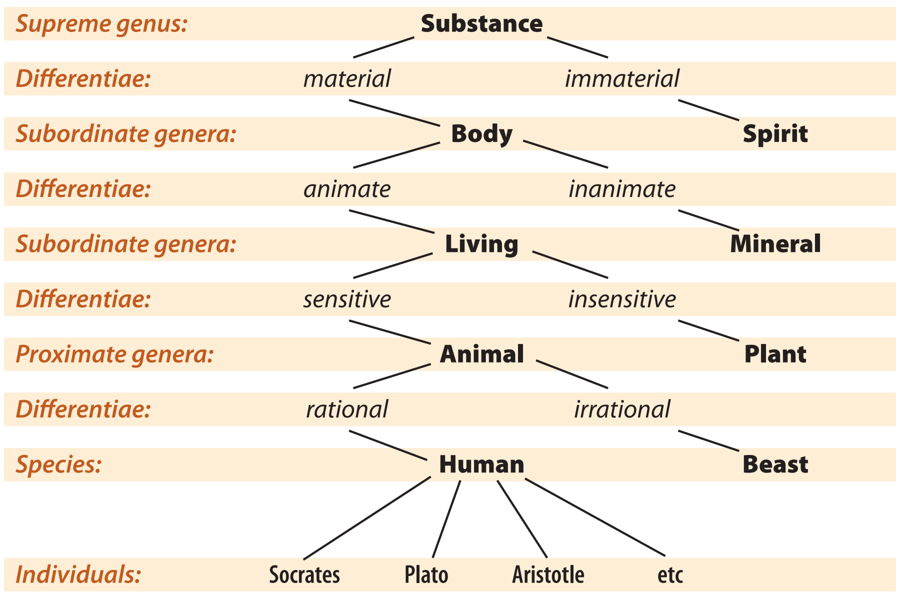
An alternative approach has been taken for the Semantic Web: instead of agreeing on a language for annotation, it has been decided to agree on **a language that can be used to specify the meaning of annotations**. This is where the idea of 'ontologies' comes in. If we can agree on an ontology language, we can use it to build up a specification of the meaning of a set of tags created for use in a given domain. New terms can then be formed by combining the existing ones; the meanings of these terms can be formally specified; and because the terms are defined in a common ontology language, terms can be combined or related across multiple ontologies.

Background to ontologies

In introducing the history of the idea of ontology, Ian reckoned computer scientists should probably apologise to philosophers for having stolen another word from them – and for having misused it into the bargain. "Well, computer scientists are always do that; we like to steal these interesting-sounding words from other domains!" he said.

Ontology, as defined by philosophers, is the most fundamental branch of metaphysics. It is concerned with being (existence), and aims to determine what entities and what *types* of entities actually exist, and thus to classify the structure of the world. Within the Western philosophical tradition, ontology was founded by the work of Plato and Aristotle. Ian showed us a diagram proposed by a later philosopher, Porphyry, and known in the Middle Ages as the *Arbor Porphyriana* [4] (See Fig. 1). 'Porphyry's Tree' presents a structure of the kinds of thing that exist, organised by what the mediaeval philosophers called *differentiae*, their distinguishing features. For example, material things that are 'bodies' can be distinguished by being either animate, in which case they are 'living', or inanimate, in which case they are 'mineral'.

Fig. 1
Porphyry's Tree



Information scientists take a more pragmatic view of the world: for them, an ontology is an engineering artefact. It consists of a vocabulary, which we use to describe a domain – or at least, some particular view of the domain. It is from this vocabulary that we can derive tags for use in a Semantic Web context. An ontology also has a mechanism for providing an explicit specification of the intended meaning of the vocabulary. This often includes

4. Porphyry (232–304 CE) was a follower of the Neoplatonist philosopher Plotinus who integrated Aristotle's work on ontology within Neoplatonism. His influential hierarchical diagram of being, the 'Arbor Porphyriana' or 'Porphyrian Tree', was taken up into Christian thinking through the works of the 6th century philosopher Boethius – and inspired the basis of later classification schemes, such as in biology.

classification-based information, not unlike what Porphyry and the other philosophers had been doing. An ontology also lets us capture constraints which give us some background knowledge of the domain.

Ian felt it was appropriate at this point to stress two factors that are essential if ontologies are going to be of any practical use:

- **An ontology should capture a shared understanding**. Those ontologies that are really proving successful in applications are those that have been developed by groups of experts working in a domain, and capture their shared understanding of the meaning of the terms that they use to describe artefacts in their domain.
- As computer scientists, we'd like our ontology to provide a **formal model** – most importantly, one that can be manipulated by machines.

Ian showed an example of an ontology built using a tool called Protégé, developed at the Stanford Medical Informatics laboratory:[5] one of the many ontology editing tools now available. The example ontology he showed (see Fig. 2 below) was about pizzas – they use it for teaching at Manchester University. The vocabulary is displayed down the left side of the screen: one of the terms there is *RealItalianPizza*. Another panel (bottom centre) shows a specification of what a real Italian pizza is – simply, a pizza whose *CountryOfOrigin* is *Italy*. The ontology also captures some background information about real Italian pizzas, which is that their base should always be *ThinAndCrispy*.

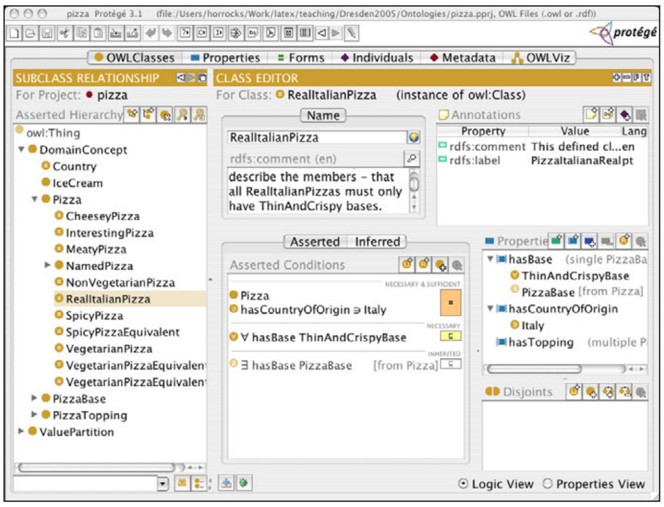


Fig. 2

A 'toy ontology' about pizzas, used in teaching at Manchester University. Shown here inside Protégé software.

Ontologies in the real world

The pizza example is, of course a 'toy ontology', but Ian was keen to stress that ontologies have a wide range of practical applications in the real world these days. In bioinformatics, several standardisation groups are building ontologies to describe important domains in biology. One example is the GO project, the Gene Ontology;[6] another is the ontology being developed by MGED, the Microarray Gene Expression Data Society.[7]

5. Protégé, written in Java and therefore runnable on any machine that provides Java 1.4 support, is a free, open-source ontology editor. See http://protege.stanford.edu/
6. The Gene Ontology — http://www.geneontology.org/
7. MGED home page – http://www.mged.org. Their ontology project: http://mged.sourceforge.net/ontologies/index.php

Biologists use these ontologies to describe information, the results of experiments and so on. Ian then showed us a diagram from an 'in silico' biology experiment, about the structure of some proteins; next to it, he displayed within a Protégé screen the ontology which was built to capture and represent knowledge about the structure of proteins in that particular protein family.

Medical practice has a very long history of using ontologies. SNOMED, NCI and GALEN are ontologies in serious day-to-day use. Whenever you see your doctor, it's likely he or she will spend much of their time with you typing stuff into the computer. At least part of what they are typing will be SNOMED codes, or information that will be translated into SNOMED codes. These codes describe both the various ailments that you may be suffering from, and also the procedures that doctors carry out. One of the drivers for the development of these ontologies is the cost of medical insurance: insurance companies use SNOMED codes to determine how much money the doctor gets for carrying out each procedure.

Ian also showed an ontology used to annotate experiments coming from visualisation of the human brain. The parts of the brain are described extremely precisely: in the sample screen he showed, some medical expert had written down in great detail exactly what the 'precentral gyrus' is, together with a wealth of background material about it. This ontology is being used, for example, in research that aims to automate the labelling of brain cortex structures in MRI images.

Many organisations are now using ontologies

Organisations such as the United Nations Food and Agriculture Organization, General Motors, Lockheed Martin and many others use ontologies to organise complex, semi-structured information. NASA has an interesting application using ontologies to organise the incredible volume of information they have about the Space Shuttle. This is a really serious problem for them because the Shuttle is such old technology now that the original designers and engineers have long since retired, making it difficult to get access to the information that they need to solve problems when they arise. Ian remembered seeing a specification for one printed circuit board with wires connecting locations; and they have to maintain several pages of regulations describing just what you are and are not allowed to do in repairing one of these wires should it break.

Ordnance Survey uses ontologies in the context of geographical information systems data (GIS). They have very detailed maps of the UK and they'd like to exploit this data in many different ways. For example, they are very interested in information about flood-plains, and they'd like to be able to annotate their map information with semantic tags that would help them to deduce where flooding is likely to occur. This is also driven, at least in part, by financial considerations, because insurance companies want to know if you live in an area where flooding is likely – and bump up your insurance premiums accordingly!

There are numerous military and government applications of ontologies. And of course there is the Semantic Web – and what is being called the 'Semantic Grid', a semantically-enhanced vision of the 'Grid' system of networked computational resources for e-Science. In these domains, ontologies are being used to provide the vocabulary for tags to annotate the data and services which are out there on the Web, and to make these computational services more accessible to automated processes.

Standardisation for the Semantic Web

The functioning of the Web depends upon standards, such as HTTP and HTML. So, if people are going to build ontologies to work in the context of the Web – where interchange and interoperability are important – the languages used to describe them will also need to be standardised. This was foreseen right at the beginning of work on the Semantic Web, and people started working on the development of suitable languages.

One of the first of these languages to emerge was RDF, and with it came **RDF Schema**, which is the metadata schema for RDF. RDF Schema is immediately recognisable as an ontology language. It talks about Classes and Properties, Sub- and Super-classes, Range and Domain and so on. But the problem with RDF Schema, said Ian, is that it's very weak: it doesn't allow you to express many things. Many features are missing – features that people building complex ontologies for use in domains like medicine, biology and so on would need. In RDF Schema, there are no constraints relating to existence or cardinality; there are no transitive, inverse or symmetrical properties; and there is no way to constrain resource descriptions by localised range or domain. Therefore there are aspects of these complex ontologies that RDF can't express.

"The other funny thing about RDF," said Ian, "is that it has a kind of 'higher-order' flavour, with not a very standard semantics, which makes it difficult to provide reasoning support."

Different groups looked at RDF Schema and they liked the basic idea, but felt that some more expressive and well-founded language would be better suited to application in the Semantic Web. A group based in Europe developed an ontology language called OIL; another group working mainly in the US developed DAML-ONT. They merged their efforts to produce 'DAML+OIL' – hardly a satisfactory name, but by this point it was clear that the project would be moving forward into a standardisation process of the World Wide Web Consortium, the W3C, which is the organisation that deals with all of the standards related to the Web, such as HTTP and XML and so on.

Within the W3C standardisation process, the project was handled by the Web Ontology Working Group, and the name that emerged was the **Web Ontology Language** – **OWL** for short. True, it perhaps should have been 'WOL' – but the team preferred the sound of 'OWL'. And as Ian pointed out, Owl in A. A. Milne's children's stories about Winnie the Pooh used to write his name WOL: "He could spell his own name WOL, and he could spell Tuesday so that you knew it wasn't Wednesday…"

OWL is now an official W3C recommendation – which means, in effect, it's a standard.

Description logics

All of these languages – OIL, DAML+OIL and OWL – are based on **description logics**, and Ian turned next to an explanation of description logics – a subject close to his heart.

Description logics are a family of logic-based knowledge representation formalisms and they are descendants of **Semantic Networks** (circa 1970) and **KL-ONE** (circa 1985), the latter being an attempt to give more formality to Semantic Networks.[8] These logics adopt an object-oriented model, similar to the model Ian had been showing us with respect to ontologies. They describe a domain in terms of classes, or 'concepts' as they are usually called in description logics; relationships between pairs of objects, which are called 'roles' in description logics; and the individuals which are instances of the classes.

The crucial advantage of description logics is that they also have operators that allow you to build complex descriptions out of simple ones that you already have. This makes it easy to extend your expressive vocabulary.

By way of explanation, Ian presented a typical description-logic example:

$$\text{HappyParent} \equiv \text{Parent} \sqcap \forall \text{hasChild}.(\text{Intelligent} \sqcup \text{Athletic})$$

In this definition of *HappyParent*, a 'happy parent' is said to be exactly equal to parents all of whose children are either intelligent or athletic. "Of course, this doesn't have to be true," said Ian, "but at least it is precisely specified! Which means, if we want to have an argument about this proposition, we know what we are talking about."

The concept of *HappyParent* is built up on the right side of the definition using class names (*Athletic*, *Intelligent*, *Parent*, *HappyParent* are all classes of individuals). The example also contains one relationship – the *hasChild* relationship. There are several operators that we use to combine these various components into one large description – we have AND, we have OR, and we also have the universal qualifier that says ALL of the children have to be either intelligent or athletic. Finally, we give a name (*HappyParent*) to this big description, and thus new vocabulary is introduced, with precise specification of what we mean by that new term in the vocabulary.

One of the things that distinguishes description logics over some earlier knowledge representation systems is that they have a formal semantics, and they really are a logic. Typically they are given a standard first-order 'model theoretic' style of semantics; in fact they are almost invariably decidable fragments of first-order logic, often contained in the C_2 (two variables with counting quantifiers) fragment. Description logics are also closely related to some other well-known decidable fragments such as Propositional Modal and Dynamic Logics, and the Guarded Fragment.

Semantics and Reasoning

To give an example of why we should be concerned with semantics and reasoning, Ian showed us an example of a semantic network: the kind of artefact that motivated work on description logics and a more precise characterisation of the semantics. The example specified an animal which is a cat and is called Felix; he is black, and sits on a mat.

To a human being, this looks fine: Felix [is-a] cat; a cat [is-a] animal; Felix [sits-on] a mat; and cat [has-colour] black. But in some respects it is ambiguous and insufficiently specified; there are several possible interpretations. Does this mean that black is the only possible colour for cats, so all cats have to be black, all over? Does it mean that cats have to have part of their colour black, but that they could have other colours as well? Or it could just mean that black is a permissable colour for cats to have, but not all cats need be black, and we

8. A good introduction to the history of development of description logics that explains the Semantic Network and KL-ONE initiatives, delves all the way back to Porphyry and forward to OWL, is John F. Sowa's paper *Semantic Networks* — http://www.jfsowa.com/pubs/semnet.htm

might have a non-black cat? The development of description logics was driven by the desire to develop this kind of object-oriented knowledge-representation language, but to pin down *exactly* what everything means in logic, and give you the opportunity to express all of these options – and to know which of them it is that you are saying.

Description logics are also distinguished from earlier knowledge representation systems in that it was always intended that they would be processed by computers. There was always the intention to provide inference services based on decision procedures for key problems; and in particular we would be able to compute satisfiability, so we could tell when some class had a consistent definition, or if it was so over-defined that you could never have an instance of that class.

In fact the intention was to implement systems that could go away and think about what we'd written down, and derive implicit information for us wsithout human guidance. If, for example, we know that John is a happy parent, and that Mary is a child of John, and Mary isn't athletic, we should be able to build a system that we can ask "Is Mary intelligent?" – and have the system answer that for us. If we were to query the system for a list of names of intelligent people, we'd like to get Mary back as part of the answer. This requires having some reasoning system that can deal with this information.

Based on years of research

Ian emphasised that the reason why we can do these things now, and the reason why a language like OWL could be built based on description logics, was because of a huge depth of existing research, more than 15 years of work, and countless papers published in the field. It is this basis that has allowed OWL to become the language that it is. The well-defined semantics for OWL (in particular, OWL DL) was inherited directly from description logics and from standard first-order semantics.

"We understand the formal properties of the description logic underlying OWL," said Ian. "We know it's actually in a quite nasty complexity class, but we know that it's decideable. We know about reasoning algorithms – and you can find papers that will tell you how to implement a program to do this kind of reasoning task I was just talking about."

There are also reasoning systems available which people can get hold of and use in their applications. One is FaCT++, a Description Logic classifier that is a development of FaCT ('Fast Classification of Terminologies') – built at Manchester, and still free. There are now also commercial reasoners such as Cerebra, a commercialisation of an earlier reasoner that Ian's team built at Manchester; and Racer, developed by STS, jointly based in Germany and Canada. (The existence of commercial software is one index of the maturity of the field.)

All this foundational research was crucial to the design of OWL, and informed the working group at every stage. Ian hinted at the kinds of debate that had gone on within the W3C Web Ontology working group. "There are always these people who take a sort of Heath Robinson approach, who tell you it's obviously harmless to extend the language with this feature or that: they would end up with a language that looked like *this*, with all kinds of bells and whistles on." And to stress the point, Ian showed a cartoon of an Heath Robinson style aeroplane with all sorts of ridiculous extras fitted (see top of next page).

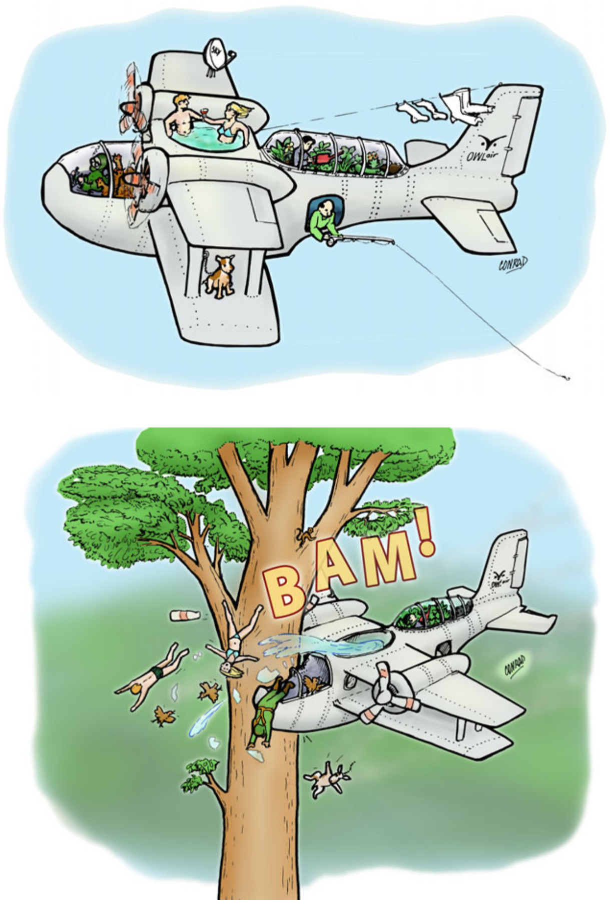


Fig 3:
What happens when you try to add too many features… (cartoons by Conrad Taylor)

This is where the many years of previous research came to the rescue. "When we were designing OWL, we were able to prevent that by being able to come back with answers such as, *No, if you have Feature X you'll crash into a big tree*." Usually the problem (the 'crash') that Feature X or Y would introduce was that it would make the language undecideable, and it was always a prerequisite of OWL that the language should be decideable.

SHOIN and OWL

Description logics have some strange names, and the description logic that underlies OWL DL is called **SHOIN**. Why SHOIN?

The reason is straightforward. Description logics are a family of formalisms, and they are mainly distinguished from each other by the operators that you have for putting things together to make descriptions like the 'happy parent' example Ian had previously shown.

- **'S'** stands for an extension of the standard basic Description Logic (known as ALC) that has been augmented with the ability to characterise certain roles (what in OWL are called 'properties') as being transitive.
- To this is added the ability to express a hierarchy not just of classes but also of roles (so, you can say things like 'having a daughter' is a sub-property of 'having a child'). This is denoted with **'H'**.
- **'O'** stands for nominals, which means you can enumerate the members of a class; for example, you can introduce a class like 'European Union' and then define that as [Italy + France + Germany +…] You can also define classes as being singletons.
- **'I'** is for Inverse Roles: e.g. you can define that 'is child of' is the reverse of 'has child'.
- **'N'** stands for Number Restrictions, which means you can define a numerical restriction on the relationships (via properties) that members of a class may have.

Constructor	Description Logic Syntax	Example	First Order Logic Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬ Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x{=}x_1 \vee \ldots \vee x{=}x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall P(x,y) \longrightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾hasChild	$\exists^{\geqslant n} y.P(x,y)$

C is a concept (class); P is a role (property); x is an individual name

Figure 4 – Class and Concept Constructors in SHOIN description logic

If you put all these letters together, you get **SHOIN**, which is therefore the name by which the logic that underpins W3C's Web Ontology Language is known.

Ian next presented a slide of 'Class/Concept Contructors', bristling with logic notation symbols (see Fig. 4 above). "You can't come to a talk by a description logic person without seeing one of these slides," he said. "These are the constructors that are available… the main interesting feature is that I can describe the whole language in this one slide." The system uses just eight constructors; it can be described very succinctly; it's very elegant; and it is compositional, so you can build complicated descriptions from these basics.

Ontology and Knowledge Bases

The term 'ontology', when used in the context of the Semantic Web, is identical to what workers in Knowledge Representation and description logics would call a **Knowledge Base**.

A Knowledge Base consists of sets of axioms. Some of the axioms describe the *structure* of concepts in a domain. In description logics, this set of axioms is typically called the **TBox** (the Terminological Box), for example 'a parent is someone who has a child'. Another set of axioms is usually collectively known as the **ABox** (the Assertional Box): these are data axioms, or 'ground facts', such as 'John is a happy parent' or 'John has a child who is Mary'. The TBox and the ABox axioms – the schema and data, if you like – combine to constitute the Knowledge Base, and an OWL ontology could be said to be not much more than a web-friendly presentation of a SHOIN Knowledge Base.

It's worth noting that when we build an ontology, unlike in a database, we don't throw away the schema when we have finished the design process. We keep the schema around and in play, because we can use it for performing interesting inferences about data. For example, we can infer that Mary is intelligent, based on the knowledge that Mary is a child of John, that John is a happy parent, and that Mary's not athletic. This inference requires us

Fig. 5

The OWL RDF/XML syntax below the line is equivalent to the assertion above the line. More verbose… but machines can process it!

$$\text{Parent} \sqcap \forall \text{hasChild}.(\text{Intelligent} \sqcup \text{Athletic})$$

```
<owl:Class>
  <owl:intersectionOf rdf:parseType="
collection">
    <owl:Class rdf:about="#Parent"/>
    <owl:Restriction>
     <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
       <owl:unionOf rdf:parseType=" collection">
           <owl:Class rdf:about="#Intelligent"/>
           <owl:Class rdf:about="#Athletic"/>
         </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

to refer back to the schema, from which we can understand the implications of the definition there of what constitutes a 'happy parent'.

The OWL RDF/XML Exchange Syntax

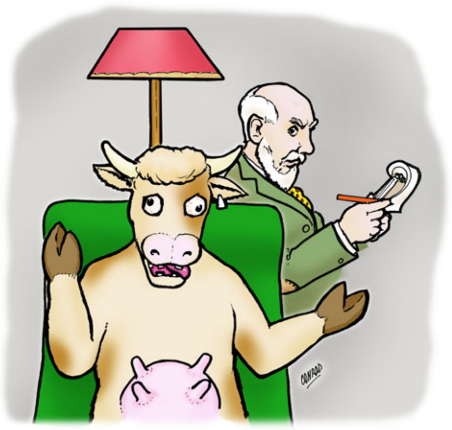
An essential task of the 3WC Web Ontology working group was to come up with a Web-friendly syntax for OWL. As the example in Figure 5 above shows, this involved translating a reasonably concise syntax into a very verbose one conforming to XML syntax.

Ian commented about the OWL version on the right, "Machines of my acquaintance assure me that they understand this extremely well. But humans – I recommend you don't show them this stuff." Yet, if you look at the example carefully, you can see that there is a clear correspondence. The class shows an intersection of *Parent*, and it has a restriction on the *hasChild* property so that all of the values are either *Intelligent* or *Athletic*.

Why ontology reasoning?

Why might we want to do reasoning with ontologies? One motivation is that, given the key and growing role of ontologies in many applications, it is going to be essential to provide tools and services to help users to design and maintain high-quality ontologies.

For example, we might want automated help so that users can decide whether the classes in an ontology they are developing are meaningful or not. By way of illustration, Ian presented an example ontology (used in his teaching) which talks about animals. Within this, there is a concept called '**mad cow**'.



The definition of a 'mad cow' is – a *cow* that eats a *brain* that's a part of a *sheep*. But if we use a reasoning tool like Pellet[9] against the ontology that includes this axiom, we are told 'this is an unsatisfiable concept'.

9. Pellet is an ontology reasoner, often used as a debugging tool, for example in the Web Ontology browser/editor 'Swoop'. See the paper Swoop: Design and Architecture of a Web Ontology Browser (/Editor) by Aditya Kalyanapur – http://www.cs.umd.edu/scholarlypapers/papers/swoop.pdf. (Note also the tendency to contrive names related the behaviour of owls!)

Why unsatisfiable? The Pellet tool will even provide an explanation: in this case it will tell you that 'mad cow' is a concept inconsistent within the ontology, because the ontology states that mad cows eat sheep brains, sheep are animals, cows are vegetarians, and vegetarians are not allowed to eat things that are either animals or parts of animals. Now, if these are your definitions, obviously you could never have a 'mad cow'! Because there is a logical contradiction present in the concept, you could never have any instances of it. Information like this is very useful for someone designing an ontology.

It's also useful to help domain experts to understand if an ontology correctly captures their intuition of the domain. Ian gave the example of a little girl, who is a domain expert in sweets and puddings. She believes that banana splits should be defined as a sub-class of fruit sundaes. This domain expert will want help from the system, to check whether or not the ontology that it computes according to these descriptions really corresponds to her own intimate knowledge of the domain. Similarly, a domain expert might want to know if there is redundant vocabulary (synonyms) – e.g. is a banana split exactly equivalent to a banana sundae? If this is the case, should the ontology contain both names?

Finally, and very importantly, we really need reasoning if we are going to carry out queries over ontology classes and instances. If we want to retrieve data that has been annotated using the ontology, such as asking for 'all intelligent people', then we need to do reasoning against the data and the schema together in order to be able to retrieve individuals like Mary, who are only *implicitly* intelligent, not explicitly annotated as such. It is performing practical tasks like this which is the *raison d'être* of the Semantic Web, after all.

A research challenge: increasing expressive power

Anyone who has worked in knowledge representation knows that users are never satisfied with the language that you give them: they always want more expressive power. OWL is quite expressive, but it has to be admitted that there are still things you can't express in OWL. For this reason there is a lot of ongoing research into how to extend the ontology language to be able to express more complex things.

For example, something that is required a great deal in complex physically-structured domains such as medicine and biology is the ability to say more complicated things about properties and their definitions. OWL gives us a rich language for building up classes, but it doesn't allow us to say much about properties yet. A well known example is the case of the *hasUncle* property, which, intuitively, we might want to define in terms of the properties *hasParent* and *hasBrother*. This is not possible yet using OWL.

In many practical applications you need to be able to talk about concrete values – integers, reals, geographical locations and so on – and you need to be able to do things with those values, such as to compare them. For example, you may want to define a class of people whose income is greater than their expenditure; and you can't do that in OWL.

People often ask for database-style keys: for example, you may want to express that the *make* plus the *model* plus the *chassis number* gives you a unique identifier for vehicles; again, you can't do that now in OWL.

A current very hot topic is how to add rule language extensions on top of OWL, and there is a newly-established W3C Working Group looking at this. There are various different flavours of rule language extension that people have been working on: for example SWRL, a Semantic Web Rule Language that extends OWL towards First Order Logic.[10] (Rule languages typically allow more expressive statements about properties – binary predicates – and would, for example, allow *hasUncle* to be defined in terms of *hasParent* and *hasBrother*.

Scaleability problems

Users also want OWL processing to go much faster. Improved scaleability is always a challenge for reasoning systems that have very high worst-case complexity, and so there is a great deal of ongoing research into optimisation techniques.

One approach suggested is reduction to **disjunctive Datalog** (a database query language), which would allow Logic Programming techniques to be used to deal with the large numbers of ground facts that are likely to be found in many practical applications. Others, including Ian himself, have suggested **hybrid DL–DB** systems that marry a Description Logic reasoner to a relational database management system to handle domains that have a very high volume of instance data. The database would be used to store all the 'Abox' axioms, derived inferences would be cached back to the database, and database queries would be used to retrieve answers to (most) logic queries: thus precomputing and caching would increase performance at query time.

There's also some very interesting work going on studying smaller description logics, in particular languages for which **polynomial-time reasoning algorithms** are available. Although these languages are much less expressive than OWL, they are adequate for many practical applications, and of course they allow for much more efficient computation.

Tools and infrastructure developments

Another active area of work is in the development of tools and infrastructure. Many editing tools are around today – for example Oiled, Protégé, Swoop, Construct and Ontotrack – and more are being developed. It is quite clear that the adoption of a standard ontology language has provided a real impetus, encouraging people to invest the effort required to build these tools. The same can be said of reasoning systems, such as Cerebra, FaCT++, Kaon2, Pellet and Racer.

Non-standard inferences is another very interesting area for development. For example, an ontology designer might want to have the system compute a 'simple' concept definition that precisely characterises a given set of individuals.

Finally, design methodologies for ontologies are also attracting a great deal of attention: in comparison with the great amount of knowledge that has been accumulated about how to set about designing a database, relatively little is yet known about how you should set about designing an ontology and what the methodology should be. In this area, work on foundational ontologies – top-level ontologies – is one potential way forward.

Ian concluded his lecture with a summary, and acknowledgements to his co-researchers, and opened the session for questions and contributions from the audience.

10. Submission to W3C re SWRL FOL — http://www.w3.org/Submission/2005/01/