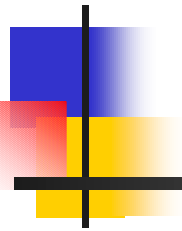




Semantic Web Lecture – Part 4

Prof. Do van Thanh

The components of the Semantic Web





Overview

- **XML** provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents.
- **XML Schema** is a language for restricting the structure of XML documents.
- **RDF** is a datamodel for objects ("resources") and relations between them. It provides a simple semantics for this datamodel. These datamodels can be represented in an XML syntax.
- **RDF Schema** is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes.



Overview

- **DAML(-ONT)**, a simple language for expressing more sophisticated RDF class definitions than permitted by RDFS.
- **DAML+OIL**, a language for expressing far more sophisticated classifications and properties of resources than RDFS and providing facilities for data typing based on the type definitions provided in the W3C XML Schema Definition Language (XSDL).
- **OWL**, derived from DAML+OIL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.



What is DAML?

- The RDF core model is successful because of its simplicity.
- The W3C also developed a purposefully lightweight schema language, RDF Schema (RDFS), to provide basic structures such as classes and properties.
- As the ambitions of RDF and XML have expanded to include things like the Semantic Web, the limitations of this lightweight schema language have become evident.
- Accordingly, the **DARPA Agent Markup Language (DAML)** program sets out to develop a more expressive schema language.
- Although DAML is not a W3C initiative, several familiar faces from the W3C, including Tim Berners-Lee, participated in its development.



What is DAML?

- In Aug 2000, **DAML-ONT**, a simple language for expressing more sophisticated RDF class definitions than permitted by RDFS is released.
- The DAML group soon pooled efforts with the **Ontology Inference Layer (OIL)** another effort providing more sophisticated classification, using constructs from frame-based AI. The result of these efforts is **DAML+OIL**, a language for expressing far more sophisticated classifications and properties of resources than RDFS.
- The most recent release was March 2001 [DAMLOIL], which also adds facilities for data typing based on the type definitions provided in the W3C XML Schema Definition Language (XSDL).
- This DAML+OIL specification and its relationship to RDF and RDFS are also available as a series of W3C notes which are used as the base specifications for the W3C **OWL**



What is ontology?

- In information science, an ontology is the product of an attempt to **formulate an exhaustive and rigorous conceptual schema about a domain.**
- An ontology is typically a **hierarchical data structure** containing all the relevant entities and their relationships and rules within that domain (e.g., a domain ontology).



What is ontology?

- However, computational ontology, like any ontology, does not have to be hierarchical at all.
- Due to the cyclical nature of the universe itself, ontology is better served with cycles.
- Hierarchy is not present in spatial contexts,
- An ontology which is not tied to a particular problem domain but attempts to describe general entities is known as a **foundation ontology** or **upper ontology**. Typically, more specialized domain specific schemata must be created to make the data useful for real world decisions.



DAML+OIL

- It gives modelers a rich expressiveness.
- It is not just a schema language but also an **ontology language**, providing primitives that support the general representation of knowledge.
- It allows one to express classifications by inference rather than by explicitly listing which resources go into which buckets.



DAML+OIL

Data typing, multiple ranges

- DAML+OIL allows property values to be restricted to the data types defined in XSDL or to user-defined data types by using a specialization of RDF properties: DatatypeProperty.

```
<daml:DatatypeProperty rdf:ID="productNumber">  
  <rdfs:label>Product Number</rdfs:label>  
  <rdfs:domain rdf:resource="#Product"/>  
  <rdfs:range  
    rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger"/>  
</daml:DatatypeProperty>
```

- The "daml" prefix represents the DAML+OIL namespace <http://www.w3.org/2001/10/daml+oil#>.



DAML+OIL

- Adds **primitives**, like `DatatypeProperty`, as needed, and it defers to RDFS otherwise.
- Some changes to the semantics of `rdfs:domain` and `rdfs:range` in DAML+OIL systems: the most important of which is that a **property** can have **multiple ranges**.
- Any property that is not a `daml:DatatypeProperty` is an `daml:ObjectProperty`, whose range must be a class defined in DAML+OIL or RDF.



DAML+OIL

- A property can also be defined as identical to another. Two ways
 - The **daml:equivalentTo**
 - Or the **daml:samePropertyAs** property

Example: Super Sports Inc. has deployed its RDF-based system using `productNumber`. A

- An online retailer consortium comes up with a standardized vocabulary for merchandise information, using uses a property called `productID` to indicate product number.
- Super Sports does not have to hack all their code to use the new property but can extend the definition of `productNumber` as follows:

```
<rdf:Description about="#productNumber">  
  <daml:samePropertyAs rdf:resource="http://consortium-of-  
    shoppers.org/vocab/productID"/>  
</rdf:Description>
```



DAML+OIL

Unique properties

- One can also specify that a property be unique, meaning that there can only be **one value of the property for each instance**

```
<daml:DatatypeProperty rdf:ID="productNumber">
  <rdfs:label>Product Number</rdfs:label>
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range
    rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger"/>
  <rdf:type
    rdf:resource="http://www.w3.org/2001/10/daml+oil#UniqueProperty"/>
</daml:DatatypeProperty>
```

Broadening the Concept of Class

- The most important facilities giving designers more expressiveness in classifying resources:

The class `daml:Class` is defined as a subclass of `rdfs:Class`, and it adds many new facilities.



DAML+OIL

- It is possible to state that a class is defined by being one of a given set of instances.

```
<daml:Class ID="Availability" >
  <daml:oneOf parseType="daml:collection" >
    <daml:Thing rdf:ID="InStock" >
      <rdfs:label>In stock</rdfs:label>
    </daml:Thing >
    <daml:Thing rdf:ID="BackOrdered" >
      <rdfs:label>Back ordered</rdfs:label>
    </daml:Thing >
    <daml:Thing rdf:ID="SpecialOrder" >
      <rdfs:label>Special order</rdfs:label>
    </daml:Thing >
  </daml:oneOf >
</daml:Class >
```



DAML+OIL

- The **daml:oneOf element** defines an enumeration, using a DAML+OIL construct that allows us to define closed lists, the `daml:collection` parse type.
- The `daml:collection` parse type allows a DAML+OIL agent to interpret the body of a property element as a special form of list, which is made up of each of the instances that appear in the element body



DAML+OIL

Disjoint classes

One class is disjoint from another if neither of the two classes have any instances in common.

Example: At Super sports, something is either a current product or a discontinued product:

```
<daml:Class rdf:ID="CurrentProduct">
  <rdfs:label>Current Product</rdfs:label>
  <rdfs:comment>An item currently sold by Super Sports Inc.
    at the time of query</rdfs:comment>
</daml:Class>
<daml:Class rdf:ID="DiscontinuedProduct">
  <rdfs:label>Discontinued Product</rdfs:label>
  <rdfs:comment>An item no longer sold by Super Sports Inc.
    at the time of query</rdfs:comment>
  <daml:disjointWith rdf:resource="#CurrentProduct"/>
</daml:Class>
```



DAML+OIL

Non-exclusive combinations

- One can also express non-exclusive boolean combinations of classes.

```
<daml:Class rdf:ID="CampingGear">  
  <rdfs:label>Camping Gear</rdfs:label>  
  <rdfs:comment>An item designed for use while camping</rdfs:comment>  
</daml:Class>
```

```
<daml:Class rdf:ID="HikingGear">  
  <rdfs:label>Hiking Gear</rdfs:label>  
  <rdfs:comment>An item designed for use while hiking</rdfs:comment>  
</daml:Class>
```

```
<daml:Class rdf:ID="FamilyProduct">  
  <rdfs:label>Family Product</rdfs:label>  
  <rdfs:comment>An item designed for family use</rdfs:comment>  
  <daml:unionOf parseType="daml:collection">  
    <daml:Class rdf:about="#CampingGear"/>  
    <daml:Class rdf:about="#HikingGear"/>  
  </daml:unionOf>  
</daml:Class>
```



DAML+OIL

- A product is a family product if and only if it is either hiking gear or camping gear. A class expression establishes a set of resources: those which are types of the class given in the expression.
- **daml:unionOf** is a standard union of the sets defined by each of the listed class expressions.

Inverse properties

- Inverse properties are quite common.
- If A is the father of B, then B is the child of A. The properties "father" and "child" are the inverse of each other.
- DAML+OIL allows one to declare this systematically, so that only assert one property, and its inverse is inferred.



DAML+OIL

Transitivity

- Another important specialization of properties in DAML+OIL is transitivity.
- For instance, the ancestor of your ancestor is also your ancestor.
- There is at least one common transitive property built into RDFS: `daml:subClassOf`.
 - If class A is a subclass of B, and class B is a subclass of C, then class A must be a subclass of C.
- DAML+OIL allows one to give this behavior to any object property one wishes

Property Restrictions

- DAML+OIL provides property restrictions, which are a way to restrict classes to a set of resources based on particular properties of theirs, the number of these properties that are asserted, or the value of these properties.



DAML+OIL

```
<daml:Class rdf:ID="MensProduct" >
  <rdfs:label>Men's Product</rdfs:label>
  <rdfs:comment>A product particularly designed to
    be used by men</rdfs:comment>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#targetSex"/>
      <daml:hasValue rdf:resource="#Male"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

- This defines the **MensProduct** class as a subclass of another class defined in-line as a DAML+OIL restriction.
- These special classes are defined by rules that specify what conditions of a resources properties must be met for that resource to be a member of the class. **daml:onProperty** identifies which property is to be checked. **daml:hasValue** then declares that the property in question must have a particular value.
- So, in effect, the above says that a men's product is a subclass of all resources which have at least one **targetSex** property whose value is **Male**.



OWL - Introduction

- **OWL Web Ontology Language** developed by the W3C Web Ontology Working Group
- Intended to be used when the **information contained in documents needs to be processed by applications**, as opposed to situations where the content only needs to be presented to humans.
- Used to explicitly represent the **meaning of terms in vocabularies and the relationships between those terms**. This representation of terms and their interrelationships is called an ontology.
- Has more facilities for **expressing meaning and semantics** than XML, RDF, and RDF-S, and higher ability to **represent machine interpretable content** on the Web.
- OWL is a **revision of the DAML+OIL** web ontology language incorporating lessons learned from the design and application of DAML+OIL.



OWL - Three sublanguages

OWL provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users:

- OWL Lite
- OWL DL
- OWL Full



OWL Lite

- Supports those users primarily needing a **classification hierarchy** and **simple constraints**.
- For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1.
- It should be simpler to provide tool support for OWL Lite than its more expressive relatives.
- OWL Lite provides a quick migration path for thesauri and other taxonomies.
- Owl Lite also has a lower formal complexity than OWL DL.



OWL - DL

- Supports those users who want the **maximum expressiveness** while retaining **computational completeness** (all conclusions are guaranteed to be computable) and **decidability** (all computations will finish in finite time).
- Includes all OWL language constructs, but they **can be used only under certain restrictions** (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class).
- OWL DL is so named due to its correspondence with **description logics**, a field of research that has studied the logics that form the formal foundation of OWL.



OWL - Full

- Is meant for users who want **maximum expressiveness** and the **syntactic freedom of RDF with no computational guarantees**.
- For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right.
- OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary.
- It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.



Relations between sublanguages

- Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded:
 - Every legal OWL Lite ontology is a legal OWL DL ontology.
 - Every legal OWL DL ontology is a legal OWL Full ontology.
 - Every valid OWL Lite conclusion is a valid OWL DL conclusion.
 - Every valid OWL DL conclusion is a valid OWL Full conclusion.



Relations between sublanguages

- Ontology developers adopting OWL should consider which sublanguage best suits their needs.
 - The choice between OWL Lite and OWL DL depends on the extent to which users **require the more-expressive constructs** provided by OWL DL.
 - The choice between OWL DL and OWL Full mainly depends on the extent to which users **require the meta-modeling facilities** of RDF Schema (e.g. defining classes of classes, or attaching properties to classes). When using OWL Full as compared to OWL DL, **reasoning support is less predictable** since complete OWL Full implementations do not currently exist.



OWL – Lite Synopsis

RDF Schema Features:

- [Class \(Thing, Nothing\)](#)
- [rdfs:subClassOf](#)
- [rdf:Property](#)
- [rdfs:subPropertyOf](#)
- [rdfs:domain](#)
- [rdfs:range](#)
- [Individual](#)

(In)Equality:

- [equivalentClass](#)
- [equivalentProperty](#)
- [sameAs](#)
- [differentFrom](#)
- [AllDifferent](#)
- [distinctMembers](#)

Property Characteristics:

- [ObjectProperty](#)
- [DatatypeProperty](#)
- [inverseOf](#)
- [TransitiveProperty](#)
- [SymmetricProperty](#)
- [FunctionalProperty](#)
- [InverseFunctionalProperty](#)



OWL – Lite Synopsis

Property Restrictions:

- [Restriction](#)
- [onProperty](#)
- [allValuesFrom](#)
- [someValuesFrom](#)

Restricted Cardinality:

- [minCardinality](#) (only 0 or 1)
- [maxCardinality](#) (only 0 or 1)
- [cardinality](#) (only 0 or 1)

Header Information:

- [Ontology](#)
- [imports](#)

Class Intersection:

- [intersectionOf](#)

Datatypes

- [xsd datatypes](#)

Versioning:

- [versionInfo](#)
- [priorVersion](#)
- [backwardCompatibleWith](#)
- [incompatibleWith](#)
- [DeprecatedClass](#)
- [DeprecatedProperty](#)

Annotation Properties:

- [rdfs:label](#)
- [rdfs:comment](#)
- [rdfs:seeAlso](#)
- [rdfs:isDefinedBy](#)
- [AnnotationProperty](#)
- [OntologyProperty](#)



OWL - DL and Full Synopsis

Class Axioms:

- [oneOf, dataRange](#)
- [disjointWith](#)
- [equivalentClass](#)
(applied to class expressions)
- [rdfs:subClassOf](#)
(applied to class expressions)

Arbitrary Cardinality:

- [minCardinality](#)
- [maxCardinality](#)

Boolean Combinations of Class Expressions:

- [unionOf](#)
- [complementOf](#)
- [intersectionOf](#)

Filler Information:

- [hasValue](#)



OWL Web Ontology Language Use Cases

- Study of usage scenarios, goals and requirements for a web ontology language.
- An ontology formally defines a common set of terms that are used to describe and represent a domain.
- Ontologies can be used by automated tools to power advanced services such as more accurate web search, intelligent software agents and knowledge management.



OWL Web Ontology Language Use Cases

- The Semantic Web is a vision for the future of the Web in which **information is given explicit meaning**, making it easier for machines to automatically process and integrate information available on the Web.
- The Semantic Web will build on XML's ability to define customized tagging schemes and RDF's flexible approach to representing data
- The next element required for the Semantic Web is a **web ontology language** which can formally describe the semantics of classes and properties used in web documents.
- In order for machines to perform useful reasoning tasks on these documents, the OWL web Ontology language must go beyond the basic semantics of RDF Schema.



OWL Web Ontology Language Use Cases

- To show the need for a web ontology language The W3C Web Ontology Working Group charter considers six representative use cases of web ontologies:
 - Web portal
 - Multimedia collections
 - Corporate web site management
 - Design documentation
 - Agents and services
 - Ubiquitous computing



Web portal ontology

- Web portals can define an ontology for the community.
- This ontology can provide a terminology for describing content and axioms that define terms using other terms from the ontology.
- For example, an ontology might include terminology such as "journal paper," "publication," "person," and "author."
- This ontology could include definitions that state things such as "all journal papers are publications" or "the authors of all publications are people."



Multimedia collections ontology

- Multimedia ontologies can be of two types:
 - media-specific
 - content-specific.
- **Media specific ontologies** could have taxonomies of different media types and describe properties of different media. For example, video may include properties to identify length of the clip and scene breaks.
- **Content-specific ontologies** could describe the subject of the resource, such as the setting or participants. Since such ontologies are not specific to the media, they could be reused by other documents that deal with the same domain.



Corporate web site management

- Large corporations typically have numerous web pages concerning things like
 - press releases
 - product offerings
 - case studies
 - corporate procedures
 - internal product briefings
 - Comparisons
 - white papers
 - process descriptions.
- Ontologies can be used to index these documents and provide better means of retrieval.



Agents and services

- The Semantic Web can provide agents with the capability to understand and integrate diverse information resources.
- Example:
 - A **social activities planner**, which can take the preferences of a user (such as what kinds of films they like, what kind of food they like to eat, etc.) and use this information to propose an **activity plan for the user** for an evening.
 - The task of planning these activities will **depend upon the richness of the service environment being offered** and the **needs of the user**.



Agents and services

- During the service determination
 - matching process
 - Ratings
 - review services
- may also be consulted to find **closer matches** to user preferences (for example, consulting reviews and rating of films and restaurants to find the "best").
- This type of agent requires **domain ontologies that represent the terms for restaurants, hotels, etc.** and service ontologies to represent the terms used in the actual services.
- These **ontologies will enable the capture of information necessary for applications to discriminate and balance among user preferences.**
- Such information may be provided by a number of sources, such as portals, service-specific sites, reservation sites and the general Web.



Design documentation

- This use case is for a **large body of engineering documentation**, such as that used by the aerospace industry.
- This documentation can be of several different types:
 - design documentation
 - manufacturing documentation
 - testing documentation.
- These document sets each have a **hierarchical structure**, but the **structures differ between the sets**.
- Ontologies can be used to build an information model which allows the exploration of the information space in terms of
 - the items which are represented
 - the associations between the items
 - the properties of the items
 - and the links to documentation which describes and defines them (i.e., the external justification for the existence of the item in the model)



Ubiquitous computing

- **Ubiquitous computing** is an emerging paradigm of personal computing, characterized by the shift from dedicated computing machinery to **pervasive computing capabilities embedded in our everyday environments**.
- Characteristic to ubiquitous computing are **small, handheld, wireless computing devices**.
- The pervasiveness and the wireless nature of devices require network architectures to support automatic, ad hoc configuration.
- A key technology of true ad hoc networks is **service discovery**, functionality by which "services" (i.e., functions offered by various devices such as cell phones, printers, sensors, etc.) can be described, advertised, discovered and used by others.
- The **utilization of services** involve discovery, contracting, and composition.



Ubiquitous computing

- An ontology language will be used to describe
 - the characteristics of devices, the means of access to such devices
 - the policy established by the owner for use of a device,
 - and other technical constraints and requirements that affect incorporating a device into a ubiquitous computing network



Support literature

- OWL Web Ontology Language Overview <http://www.w3.org/TR/2004/REC-owl-features-20040210/>
- OWL Web Ontology Language Use Cases and Requirements, Jeff Heflin, Editor, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-webont-req-20040210/> . Latest version available at <http://www.w3.org/TR/webont-req/> .



Bibliography

- Extensible Markup Language (XML). <http://www.w3c.org/XML/>
- XML Schema . <http://www.w3c.org/XML/Schema>
- XML Schema Part 2: Datatypes - W3C Recommendation, World Wide Web Consortium, 2 May 2001.
- RDF/XML Syntax Specification (Revised), Dave Beckett, Editor, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/> . Latest version available at <http://www.w3.org/TR/rdf-syntax-grammar/> .
- Resource Description Framework (RDF): Concepts and Abstract Syntax, Graham Klyne and Jeremy J. Carroll, Editors, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/> . Latest version available at <http://www.w3.org/TR/rdf-concepts/>
- RDF Vocabulary Description Language 1.0: RDF Schema, Dan Brickley and R. V. Guha, Editors, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/> . Latest version available at <http://www.w3.org/TR/rdf-schema/> .
- RDF Semantics, Patrick Hayes, Editor, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/> . Latest version available at <http://www.w3.org/TR/rdf-mt/> .
- The Description Logic Handbook. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel-Schneider, editors. Cambridge University Press, 2003; and Description Logics Home Page.



Bibliography

- DAML+OIL Reference Description . Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. W3C Note 18 December 2001.
- DAML. <http://www.daml.org>



Bibliography

- OWL Web Ontology Language Overview <http://www.w3.org/TR/2004/REC-owl-features-20040210/>
- OWL Web Ontology Language Guide, Michael K. Smith, Chris Welty, and Deborah L. McGuinness, Editors, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-owl-guide-20040210/> . Latest version available at <http://www.w3.org/TR/owl-guide/> .
- OWL Web Ontology Language Reference, Mike Dean and Guus Schreiber, Editors, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-owl-ref-20040210/> . Latest version available at <http://www.w3.org/TR/owl-ref/> .
- OWL Web Ontology Language Semantics and Abstract Syntax, Peter F. Patel-Schneider, Pat Hayes, and Ian Horrocks, Editors, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-owl-semantics-20040210/> . Latest version available at <http://www.w3.org/TR/owl-semantics/> .
- OWL Web Ontology Language Test Cases, Jeremy J. Carroll and Jos De Roo, Editors, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-owl-test-20040210/> . Latest version available at <http://www.w3.org/TR/owl-test/> .
- OWL Web Ontology Language Use Cases and Requirements, Jeff Heflin, Editor, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-webont-req-20040210/> . Latest version available at <http://www.w3.org/TR/webont-req/> .
- Web Ontology Issue Status. Michael K. Smith, ed. 1 November 2003.