Tutorial: Constructing Ontologies for Semantic Web Applications

Jurika Shakya, Marek Hatala, Vive Kumar
shakya@sfu.ca
Simon Fraser University

Friday, November 19, 2004
Tutorial Outline

- Overview
- Developing Sample ontology (in Protégé)
- Ontology Research in SFU Surrey
- Demo of Echo
Goal of this tutorial

- Semantic Web and Ontology
- Understand OWL and its elements
- Able to build an ontology (in Protégé)
- Know some of the limitations and common mistakes in building ontology
- Application and Use of ontology
1. Overview
Ontology Overview

- Ontology is one of the layer in the foundation for Semantic Web

- What is Semantic Web?
  - the Semantic Web, semantic indicates that the meaning of data on the Web can be discovered—not just by people, but also by computers.
  - The purpose of semantic web is to make semantic content accessible to machines and not humans alone.

- Ontologies provide the meaning in the Semantic Web
  - Ontologies provide a vocabulary of terms
  - New terms can be formed by combining existing ones
  - Meaning (semantics) of such terms is formally specified
  - Can also specify relationships between terms in multiple ontologies
Ontology Overview

What is ontology?

- An ontology is an explicit specification of a conceptualization [Gruber93].

- An ontology is an engineering artifact:
  - It is constituted by a specific vocabulary used to describe a certain reality, plus
  - a set of explicit assumptions regarding the intended meaning of the vocabulary. [Horrocks]

- Thus, an ontology describes a formal specification of a certain domain:
  - Shared understanding of a domain of interest
  - Formal and machine manipulable model of a domain of interest

- Ontology is a catalog of the types of things that are assumed to exist in a domain of interest $D$ from the perspective of a person who uses a language $L$ for the purpose of talking about $D$ [Sowa99]
Ontology Overview

Why Ontology?

- Concepts (classes, entities)
- Properties of concepts (slots, attributes, roles)
- Relationships between concepts (associations)
- Additional Constraints (cardinality, type, encapsulation)

These are not all supported by all ontology language or representation. However, these are some of the core of elements of ontology.
Ontology Overview

How to implement Ontology?

- There are wide variety of ontology languages and representation
  - **Graphical notations**
    - Semantic networks (see http://www.jfsowa.com/pubs/semnet.htm)
    - Topic Maps (see http://www.topicmaps.org/)
    - UML (see http://www.uml.org/)
    - RDF (see http://www.w3.org/RDF/)
  - **Logic based**
    - Description Logics (e.g., OIL, DAML+OIL, OWL)
    - Rules (e.g., RuleML, LP/Prolog)
    - First Order Logic (e.g., KIF)
    - Conceptual graphs (e.g. CGs. See http://www.jfsowa.com/cg/)
    - (Syntactically) higher order logics (e.g., LBase)
    - Non-classical logics (e.g., Flogic, Non-Mon, modalities)
  - **Probabilistic/fuzzy**
    - OWL can be extended to incorporate probabilistic/Fuzzy logic
Ontology Overview

- **Requirements for Ontology Language:**
  - Extends existing Web standards (e.g., RDF, RDFS)
  - Easy to understand and use
  - Should be based on familiar KR idioms
  - Formally specified
  - Of “adequate” expressive power
  - Possible to provide automated reasoning support

- Two languages were developed to satisfy above requirements
  - **OIL**: developed by group of (largely) European researchers (several from EU OntoKnowledge project)
  - **DAML-ONT**: developed by group of (largely) US researchers (in DARPA DAML programme)

- These two languages were merged to form **DAML+OIL** and presented to w3C

- **OWL** standard language (Feb 2004) based on predecessor DAML+OIL
OWL Overview

- OWL is a language and framework for representing ontological knowledge and information about the way that the world is structured and fits together.

- There are **three different variants** of OWL implementation
  - OWL Full
    - based on the union of OWL Syntax and the RDF
  - OWL DL *(will be the focus the rest of the tutorial)*
    - Part of OWL Full that fits in the DL Framework
    - Has decidable reasoning (DL semantics officially definitive)
  - OWL Lite
    - A subset of OWL DL
    - Easier reasoning but least expressive

- OWL has specific characteristics such as
  - Based on KR idioms (DL)
  - Extends web standards such as XML, RDF, RDFs
  - Product of research done on DAML, OIL, DAML+OIL
  - Expressive power
  - Possible to provide automated reasoning support
OWL Overview

Components of OWL ontologies:

- **Individuals**
  - Represent objects in the domain of discourse
  - Also known as *Instances, constants in FOL*
  - *Note:* Two names could represent the same “real-world” individual as OWL does not use Unique Name Assumption (UNA)

- **Properties**
  - Binary relations on individuals
  - A collection of relationships between individuals
  - Properties can have inverse properties
  - Can be functional (single value)
  - Can be transitive or symmetric
  - Also known as *slots, attributes (some formalisms, relations (UML), roles (DL)*
OWL Overview

Components of OWL ontologies:

- **Classes**
  - Sets of individuals with common characteristics that is formally described
  - Can be arranged as super class-subclass hierarchy, which is also known as a taxonomy
  - Also known as concepts, category, type
  - *Note*: Classes can overlap arbitrarily. If the two classes should not overlap then need to formally set them as disjoint.
## DL Components of OWL ontologies:

### OWL Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>Modal Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
<td>$C_1 \land \ldots \land C_n$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
<td>$C_1 \lor \ldots \lor C_n$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>Male</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \sqcup \ldots \sqcup {x_n}$</td>
<td>${\text{john}} \sqcup {\text{mary}}$</td>
<td>$x_1 \lor \ldots \lor x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall \text{hasChild}.\text{Doctor}$</td>
<td>$[P]C$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists \text{hasChild}.\text{Lawyer}$</td>
<td>$\langle P \rangle C$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq nP$</td>
<td>$\leq 1\text{hasChild}$</td>
<td>$[P]_{n+1}$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq nP$</td>
<td>$\geq 2\text{hasChild}$</td>
<td>$\langle P \rangle_n$</td>
</tr>
</tbody>
</table>
OWL Overview

DL Components of OWL ontologies:

### OWL Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\sqcap$ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\sqcap$ Male</td>
</tr>
<tr>
<td>disjointWith</td>
<td>$C_1 \sqsubseteq \neg C_2$</td>
<td>Male $\sqsubseteq \neg$ Female</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>${x_1} \equiv {x_2}$</td>
<td>${\text{President Bush}} \equiv {\text{G W Bush}}$</td>
</tr>
<tr>
<td>differentFrom</td>
<td>${x_1} \subseteq \neg {x_2}$</td>
<td>${\text{john}} \subseteq \neg {\text{peter}}$</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \equiv P_2^-$</td>
<td>hasChild $\equiv$ hasParent$^-$</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor$^+$ $\sqsubseteq$ ancestor</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>$\top \sqsubseteq \preceq 1P$</td>
<td>$\top \sqsubseteq 1\text{hasMother}$</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>$\top \sqsubseteq \preceq 1P^-$</td>
<td>$\top \sqsubseteq 1\text{hasSSN}$</td>
</tr>
</tbody>
</table>
OWL Overview

Components of OWL ontologies:

- DL mainly characterized by
  - Set of constructors for building concepts and roles
  - Set of axioms for asserting facts about concepts, roles and individuals

- Benefits of Description Logic
  - Logic allows the axiomatization of the domain information through
    - Syntax
    - Semantics
  - Logical inference or Reasoning
2. Ontology Editors
Ontology Editors

- Protégé
- OilEd
- Ontolingua
- OntoEdit
- WebOnto
- etc…

- coming soon…LORNET’s own ontology editor.
Protégé and OWL plugin Architecture

Jena API
(Parsing, Reasoning)

OWL File Storage

Protégé Core System

Protégé API
(Classes, properties, individuals, etc.)

Protégé OWL API
(Logical class definitions, restrictions, etc.)

Protégé OWL GUI
(Expression Editor, Conditions Widget, etc.)

Protégé GUI
(Tabs, Widgets, Menus)

OWL Plugin

OWL Extension APIs
(SWRL, OWL-S, etc.)

OWL GUI Plugins
(SWRL Editors, ezOWL, OWLViz, Wizards, etc.)

DB Storage

OWL File Storage

Protégé and OWL plugin Architecture
Developing Ontology

Requirements:

- Protégé 2.1.1 or later ([http://protege.stanford.edu/](http://protege.stanford.edu/)) (current version 3.0)
- Protégé-OWL plugin (latest) ([http://protege.stanford.edu/plugins/owl/](http://protege.stanford.edu/plugins/owl/))
- Racer (latest) ([http://www.sts.tu-harburg.de/~r.f.moeller/racer/](http://www.sts.tu-harburg.de/~r.f.moeller/racer/))

These are the basics, most are bundled with the latest Protégé download, except for Racer.
Ontology Development Cycle

- Ontology development is an iterative process

- Ontology development requires
  - some experience and foresight
  - communication between domain experts and developers
Developing Ontology

Tutorial Sample

- Domain: Education Technology
- Scope: Teaching Strategies and Tactics
- Goal: Find Matching Strategy for Instructor expectation
- Problem: What is the teaching strategy to use with potential content, for a particular student,...
Developing Ontology

- Tutorial Sample
  - Any existing Ontology?
  - ka.owl
    (http://protege.stanford.edu/plugins/owl/ontologies.html/ka.html)
    - Defines concepts from academic research.
    - Contributed by Ian Horrocks
Developing Ontology

- Tutorial Sample
  - Architecture

- Required Models
  - Educational Theories
  - Teaching Strategies
  - Teaching Tactics
Developing Ontology

Creating Classes

- Create a new OWL Project, selecting **OWL Files**
- Create subclasses to owl:Thing, by clicking 🔄
- To specify that classes are not overlapping, we need to set them to be disjoint
- You can also use OWL Wizard to create Group of subclasses.
Create OWL Project
Create simple class
Create group class using wizard
Create class hierarchy and set disjoints
Developing Ontology

Creating Properties

- Create DataType Property (simple values) for a class
  - Attaches properties to a class.

- Create ObjectType Property (references)
  - Attaches properties between instances

- Create Annotation Property (add additional information)
  - Attaches additional non-inferable information
Create DataType Property
Create ObjectType Property
Developing Ontology

OWL Property Characteristics

- Functional Properties (single valued property)
- Inverse Functional Properties (inverse property is functional)
- Transitive Properties
- Symmetric Properties

Property Domain and Range

- Usually Domains and Range does **not** behave as constraints and they can cause ‘unexpected' classification results
- It is **incorrect** to specify a class rather than its individuals as range of property
Developing Ontology

- Defining Classes

- Property Restrictions (Quantifier, Cardinality, hasValue)

- Primitive vs. Defined Class
  - A class that only has *necessary conditions* is known as a Primitive Class.
  - A class that has at least one set of *necessary and sufficient conditions* is known as a Defined Class.
Defining Classes - example of property restriction
Primitive vs. Defined
Developing Ontology

- Importing other ontology

- Protégé OWL allows importing of one or more external ontology
- Upper Ontologies can be imported through the co-ordination of the namespaces.
- The ontology to be imported is pointed out to the location by the use of namespace URI
Importing Other Ontology
Importing Other Ontology
Developing Ontology

- Testing ontology - Setup the test settings
Developing Ontology

- Testing ontology – Running the test
Developing Ontology

- Using Reasoner*

  - Use Species validation facility to decide on the OWL sub-language
  - Classifying Taxonomy
  - Check consistency

*Note: Have to have Racer or some other reasoner running in the background. Has to be connected to the internet.
RACER Version 1.7.21
RACER: Reasoner for Aboxes and Concept Expressions Renamed
Supported description logic: ALCQHIr+(D)-
Copyright (C) 1998-2004, Ulinker Haarslev and Ralf Möller.
Commercial use is prohibited; contact the authors for licensing.
RACER is running on IBM PC Compatible computer as node Unknown
The XML/RDF/RDFS/DAML parser is implemented with Wilbur developed
by Ora Lassila. For more information on Wilbur see
The store/restore facility is based on software developed
by Michael Wessel.
The solver for nonlinear inequations over the complex numbers
is based on CGB by Marek Rychlik, University of Arizona.
For more information on CGB see http://alamos.math.arizona.edu/~rychlik/.
The HTTP interface based on DIG is implemented with CL-HTTP developed and
owned by John C. Mallery. For more information on CL-HTTP see
[2004-07-30 01:14:35] TCP service enabled on port 8088
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rss="http://purl.org/rss/1.0/
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
    xmlns="http://protege.stanford.edu/plugins/owl/owl-library/ka.owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xml:base="http://protege.stanford.edu/plugins/owl/owl-library/ka.owl">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:ID="PaperLibrary">
        <rdfs:subClassOf>
            <owl:Class rdf:ID="KAMethodology"/>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Activity">
        <rdfs:subClassOf>
            <owl:Class rdf:ID="Event"/>
        </rdfs:subClassOf>
    </owl:Class>
    ...

OWL file
Approaches

- Design patterns in developing Ontology
  - Value Partitions
    - Not part of OWL or any other ontology languages
    - Can be used for refining class description
    - In Protégé, can create value partition through the OWL Wizard tool
    - Not many available at the moment, maybe in future tools

- Common errors & misconceptions
  - Restrictions using “allValuesFor” instead of “someValuesFor”
    - When in doubt, use “SomeValuesFrom”
  - Forgetting to set “Defined class” instead of “Primitive class”
  - OWL is Open World
    - Everything is possible until we say it is impossible
  - In OWL, domain & range constraints are equivalent to “allValuesFrom” restrictions
  - Confusing kind-of & part-of in constructing the tree.
## References:

<table>
<thead>
<tr>
<th>Reference</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Rector. et al.02]</td>
<td>OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors &amp; Common Patterns</td>
</tr>
<tr>
<td>[Knublauch04]</td>
<td>Editing OWL Ontologies with Protégé, 7th Protégé Conference Workshop, Washinton, DC, 2004</td>
</tr>
</tbody>
</table>
## References:

<table>
<thead>
<tr>
<th>Reference</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Medvedeva et.al. 04]</td>
<td>Reasoning in a Tutoring System: Transforming Knowledge to Teaching, Protégé 2004</td>
</tr>
<tr>
<td>[Pool et.al. 04]</td>
<td>KEEPER and Protégé: An Elicitation Environment for Bayesian Inference Tools, Workshop on Protégé and Reasoning, 2004</td>
</tr>
<tr>
<td>[Wang et.al 04]</td>
<td>Case Study in Rule Inferencing using Protégé and Jess</td>
</tr>
<tr>
<td>[Kumar et. al 04]</td>
<td>Toward An Ontology of Teaching Strategies, ITS2004, Brazil, 2004</td>
</tr>
</tbody>
</table>
Walkthrough on Travel Destination Ontology
(developed by Holger Knublauch)
Tutorial Scenario

- Semantic Web for Tourism/Traveling
- Goal: Find matching holiday destinations for a customer

I am looking for a comfortable destination with beach access

Tourism Web
Scenario Architecture

- A search problem: Match customer’s expectations with potential destinations
- Required: Web Service that exploits formal information about the available destinations
  - Accommodation (Hotels, B&B, Camping, ...)
  - Activities (Sightseeing, Sports, ...)

Outline
Tourism Semantic Web

- Open World:
  - New hotels are being added
  - New activities are offered
- Providers publish their services dynamically
- Standard format / grounding is needed → Tourism Ontology
Tourism Semantic Web

- OWL Metadata (Individuals)
- OWL Metadata (Individuals)
- OWL Metadata (Individuals)
- OWL Metadata (Individuals)

Tourism Ontology

- Destination
- Activity
- Accommodation

Web Services
Cardinality Restrictions

- Meaning: The property must have at least/at most/exactly x values
-  is the shortcut for  and  
- Example: A FamilyDestination is a Destination that has at least one Accommodation and at least 2 Activities
**hasValue Restrictions**

- **Meaning:** At least one of the values of the property is a certain value

- **Similar to someValuesFrom** but with Individuals and primitive values

- **Example:** A PartOfSydney is a Destination where one of the values of the isPartOf property is Sydney
Enumerated Classes
- Consist of exactly the listed individuals

- OneStarRating
- TwoStarRating
- ThreeStarRating

Asserted Conditions
- Accomodation
- ∃ hasRating {OneStarRating, TwoStarRating}

BudgetAccomodation
Classification

NationalPark

- A RuralArea is a Destination
- A Campground is BudgetAccomodation
- Hiking is a Sport

Therefore:
Every NationalPark is a Backpackers-Destiantion

BackpackersDestination

- Destination
- ∃ hasAccomodation BudgetAccomodation
- ∃ hasActivity (Sports △ Adventure)

(Other BackpackerDestinations)
Classification (2)

- Input: Asserted class definitions
- Output: Inferred subclass relationships
Outline Visualization with OWLViz
Outline

OWL Wizards

Create Value Partition

Properties Matrix

Steps
- Add Values
- Verify Values
- Add Annotations
- Select Value Partitions Root
- Finish!

Steps
- Choose Classes
- Choose Properties
- Set Properties
- Finish!
Ontology Research in SFU Surrey

at the LOR (Laboratory for Ontology Research)
Research Projects

- Jurika Shakya – A framework for curriculum model (student model, courseware model, administrative model, cost model, workflow, …)
- David Brokenshire and Brittney Bogyo – Ontology for information flow between ePortfolio and User model
- Jon Hatol – Semantic aggregation of Dance Notation learning objects in Labanotation
Research Projects

- Kate Han – Estimation of Quality of Learning Objects using Distributed BBN
- Wenting Ma, Stephanie Chu – Ontology for instructional design
- Timmy Eap – Course Ontology
- Rob Woodbury – A.VI.RE Ontologies
- Jerry Li – Ontology for eLera
Research Projects

- Leila Kalantari - Echo Project – Ontology for Museum visits
- Andrew Choi – Wordnet signatures in mapping ontologies at a distance
- April Ng – Competency ontology for recommendation of learning resources
- Derek Kingston and Timmy Eap – Ontology for mapping people to LOR registry
- Jerry Li – Mapping Local Ontologies for eLera
Demo of Echo

Knowledgebase development and Reasoning using JESS
THANK YOU