A Language for Ontology Engineering: OWL and Rules

Vilas Wuwongse
Computer Science and Information Management Program
School of Advanced Technologies
Asian Institute of Technology, Thailand
Email: vw@cs.ait.ac.th
Contents

- Ontology
- Ontology Engineering
- Ontology Languages (OWL and OWL with Rules)
- XDD Modeling of OWL and Rules
- Conclusions
Ontology
Definitions of Related Terms

- A **controlled vocabulary** is a list of terms that have been enumerated explicitly and controlled by a controlled vocabulary registration authority.

- A **taxonomy** is a collection of controlled vocabulary terms organized into a hierarchical structure. Each term in a taxonomy is in one or more parent-child relationships to other terms in the taxonomy. [ETYMOLOGY: 19th Century: from French *taxonomie*, from Greek *taxis* order + -nomy]

- A **thesaurus** is a networked collection of controlled vocabulary terms including their synonyms and related terms, i.e., a thesaurus uses associative relationships in addition to parent-child relationships. [ETYMOLOGY: 18th Century: from Latin, Greek: treasure]
Definition of Ontology

- Webster’s Definition
  1: a branch of metaphysics concerned with the nature and relations of being
  2: a particular theory about the nature of being or the kinds of existents

- The word ontology is from the Greek *ontos* for being and *logos* for word.

- People use the word *ontology* to mean different things, e.g. glossaries & data dictionaries, thesauri & taxonomies, schemas & data models, and formal ontologies & inference.
Ontology in Computer Science


An ontology is

- *a formal, explicit specification of a shared conceptualization* [Gruber93]
- *a common vocabulary and agreed upon meanings to describe a domain of interest*

Meanings of the keywords:

- *conceptualization*: abstraction of some real-world phenomenon
- *shared*: acceptance by a community, not restricted to some individuals
- *specification*: definition
- *explicit*: crystal-clear declarative meaning
- *formal*: machine-processability

In short, an ontology provides

- *a common vocabulary* of terms
- declarative definition of the *meaning of the terms* (semantics)
- a *shared understanding* for people as well as machines
Object-Concept-Representation

[Sowa]
Meta-Concept

[Diagram showing the concept of representation with categories: Object, Concept, Symbol, Symbol of Concept, and Representation (Repr).]
Ontology Engineering
Definition

- **Ontology Engineering** deals with the representation, design, development, implementation and application of ontologies.
- Ontology Engineering tasks include:
  - Conceptualization
  - Determination of concept relationships, axioms and constraints
  - Formulation of ontology representation
  - Implementation of the ontology representation in an ontology language
  - Application of the ontology
Conceptualization Techniques

- Formal Concept Analysis (FCA)

<table>
<thead>
<tr>
<th>Concept Types</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nonalcoholic</td>
</tr>
<tr>
<td>HerbTea</td>
<td>x</td>
</tr>
<tr>
<td>Coffee</td>
<td>x</td>
</tr>
<tr>
<td>MineralWater</td>
<td>x</td>
</tr>
<tr>
<td>Wine</td>
<td></td>
</tr>
<tr>
<td>Beer</td>
<td></td>
</tr>
<tr>
<td>Cola</td>
<td>x</td>
</tr>
<tr>
<td>Champagne</td>
<td></td>
</tr>
</tbody>
</table>

Table of beverage types and attributes

[Sowa]
Conceptualization of a Water Ontology

NaturallyOccurringWaterSource

Stream

Brook

River

Tributary

BodyOfWater

Lake

Ocean

Sea

Rivulet

[Costello and Jacobs]
Determination of Water Ontology Properties

- NaturallyOccurringWaterSource
  - Stream
    - Brook
    - Rivulet
  - River
  - Tributary
  - Lake
  - Ocean
    - Sea

Properties:
- connectsTo: NaturallyOccurringWaterSource (Symmetric)
- feedsFrom: River (Inverse Functional)
- emptiesInto: BodyOfWater (Functional)
- containedIn: BodyOfWater (Transitive)

[Costello and Jacobs]
Ontology Languages
OWL (Web Ontology Language)

- OWL is an XML vocabulary that is used to define classes, their properties, as well as class and property relationships.
- OWL can define
  - Classes
  - Properties
  - Individuals
  - Subclass and other types of class relationships
  - Property relationships
  - Restrictions for property values
  - Individual relationships
- OWL is an extension of RDFS (Resource Description Framework Schema)
- OWL enables machine-processable semantics.
Examples of OWL Vocabulary

- **subClassOf** asserts that one class of items is a subset of another class of items
- **equivalentProperty** asserts that one property is equivalent to another
- **sameIndividualAs** asserts that one instance is the same as another instance
- **maxCardinality** specifies the maximum number of objects satisfying a property
Camera Ontology

Camera

SLR
- Properties:
  - viewFinder: hasValue = #ThroughTheLens
    (SLR is characterized by a viewFinder that is through the lens)

Digital

Large-Format
- Properties:
  - body: allValuesFrom
    BodyWithNonAdjustableShutterSpeed
    (i.e., no shutter-speed adjusting on LF cameras)

[Costello and Jacobs]
Example of using OWL to define two terms and their relationship

Example: Define the terms "Camera" and "SLR". State that SLRs are a type of Camera.

These two terms (classes) and their relationship is defined using the OWL vocabulary

```xml
<owl:Class rdf:ID="Camera"/>
<owl:Class rdf:ID="SLR">
  <rdfs:subClassOf rdf:resource="#Camera"/>
</owl:Class>
```

[Costello and Jacobs]
Relationship between focal-length and lens size

This OWL element states that focal-length is equivalent to lens size.

```
<owl:DatatypeProperty rdf:ID="focal-length">
  <owl:equivalentProperty rdf:resource="#size"/>
  <rdfs:domain rdf:resource="#Lens"/>
  <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>
```

"focal-length is synonymous with (lens) size"

[Costello and Jacobs]
Summary of OWL Vocabulary: Class Constructors

- **allValuesFrom**: \( P(x, y) \) and \( y = \text{allValuesFrom}(C) \)
- **someValuesFrom**: \( P(x, y) \) and \( y = \text{someValuesFrom}(C) \)
- **cardinality**: \( \text{cardinality}(P) = N \)
- **minCardinality**: \( \text{minCardinality}(P) = N \)
- **maxCardinality**: \( \text{maxCardinality}(P) = N \)
- **intersectionOf**: \( C = \text{intersectionOf}(C_1, C_2, ...) \)
- **unionOf**: \( C = \text{unionOf}(C_1, C_2, ...) \)
- **complementOf**: \( C = \text{complementOf}(C_1) \)
- **oneOf**: \( C = \text{one of}(v_1, v_2, ...) \)

where:

- \( C, C_1, C_2 \): OWL descriptions
- \( P \): an OWL property
- \( x, y \): variables, OWL individuals or OWL data values
- \( N \): a number
Summary of OWL Vocabulary: Axioms

\textbf{subClassOf}: \( C_1 = \text{subClassOf}(C_2) \)

\textbf{equivalentClassOf}: \( C_1 = C_2 \)

\textbf{disjointWith}: \( C_1 \neq C_2 \)

\textbf{transitiveProperty}: if \( P(x,y) \) and \( P(y,z) \) then \( P(x, z) \)

\textbf{FunctionalProperty}: if \( P(x,y) \) and \( P(x,z) \) then \( y=z \)

\textbf{InverseOf}: if \( P_1(x,y) \) then \( P_2(y,x) \)

\textbf{InverseFunctionalProperty}: if \( P(y,x) \) and \( P(z,x) \) then \( y=z \)

\textbf{equivalentProperty}: \( P_1 = P_2 \)

\textbf{subPropertyOf}: \( P_1 = \text{subClassOf}(P_2) \)

\textbf{equivalentPropertyOf}: \( P_1 = P_2 \)

\textbf{sameIndividualAs}: \( I_1 = I_2 \)

\textbf{differentFrom}: \( I_1 \neq I_2 \)

where:

- \( C, C_1, C_2 \): OWL descriptions
- \( P_1, P_2 \): OWL properties
- \( x, y, z \): variables, OWL individuals or OWL data values
- \( I_1, I_2 \): individuals
Summary of OWL Vocabulary: Axioms

- **subClassOf**: C1 = subClassOf(C2)
- **equivalentClassOf**: C1 = C2
- **disjointWith**: C1 != C2
- **transitiveProperty**: if P(x,y) and P(y,z) then P(x, z)
- **FunctionalProperty**: if P(x,y) and P(x,z) then y=z
- **InverseOf**: if P1(x,y) then P2(y,x)
- **InverseFunctionalProperty**: if P(y,x) and P(z,x) then y=z
- **equivalentProperty**: P1 = P2
- **subPropertyOf**: P1 = subClassOf(P2)
- **equivalentPropertyOf**: P1 = P2
- **sameIndividualAs**: I1 = I2
- **differentFrom**: I1 != I2

---

where:

C, C1, C2: OWL descriptions

P1, P2: OWL properties

x, y, z: variables, OWL individuals or OWL data values

I1, I2: individuals
OWL with Rules

- In order to extend the expressive power of OWL, a Semantic Web Rule Language (SWRL) has been proposed.
- SWRL combines OWL DL and OWL Lite sublanguages of OWL with the Unary/Binary Datalog sublanguages of RuleML (http://www.ruleml.org), enabling Horn-like rules to be combined with an OWL knowledge base.
- The proposed rules are of the form of an implication between an antecedent (body) and consequent (head), where both the antecedent and consequent consist of zero or more atoms.
- The atoms can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values.
Example of SWRL

This rule asserts that if x1 hasParent x2, x2 hasSibling x3, and x3 hasSex male, then x1 hasUncle x3.
XDD Modeling of OWL + Rules
XDD is unified, XML-based knowledge representation language with well-defined declarative semantics, and a support for general computation and inference mechanisms.

It employs:
- XML’s nested tree structure as its underlying data structure,
- Declarative Description theory as a framework to enhance its expressive power.
Problems with SWRL

- SWRL is a mere XMLization of a subset of Horn logic
- SWRL is too verbose and is a not succinct representation of real-world domain data
- Handling of XML data by SWRL is not direct
- Efficient computational mechanism may be difficult to develop
XDD Descriptions

An XDD Description

- Ordinary XML Elements
  - Representing explicit information items in a particular domain and denoting a semantic unit

- XML Expressions
  - Representing implicit information or a set of semantic units
  - (Extended XML Elements with Variables)

- XML Clauses
  - Modeling integrity constraints, rules, conditional relationships and axioms
XML Clauses

H

← B₁ ,

B₂ ,

. .

Bₙ .

Head

Body
A description of domain-specific ontologies and their instances encoded in an ontology language, such as OWL, becomes immediately an XDD description comprising solely ordinary XML elements.

XML clauses can be employed to define the axiomatic semantics of each ontology modeling primitive which includes a certain notion of implication.

XML clauses can be used to model arbitrary rules, axioms, constraints and queries.
XDD Description:
Ontologies and instances

C1: <owl:Class rdf:ID="Person">
    <rdfs:label>person</rdfs:label>
</owl:Class>

C2: <owl:ObjectProperty rdf:ID="hasChild">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>

C3: <owl:ObjectProperty rdf:ID="hasParent">
    <owl:inverseOf rdf:resource="#hasChild"/>
</owl:ObjectProperty>

C4: <Person rdf:about="Jack">
    <age>52</age>
    <hasChild rdf:resource="#John"/>
    <hasAirlineMembership/>
</Person>

C5: <Person rdf:about="John">
    <age>29</age>
    <hasChild rdf:resource="#Jill"/>
    <hasAirlineMembership rdf:resource="#tg9000"/>
</Person>

C6: <Person rdf:about="Jill">
    <age>7</age>
    <hasAirlineMembership/>
</Person>

Application-specific ontology definition expressed in terms of OWL.

Ontology instances (application data)
XDD Description: Ontology Axioms

If a property R is an inverse of a property P, then for any resource X the value of a property P of which is a resource Y, one can infer that Y also has a property R the value of which is the resource X.

C7: `<$N:classB rdf:about=$S:resourceY>
    <$E:instance1Elmt>
    <$$S:propertyR rdf:resource=$S:resourceX/>
</$N:classB>
←
<owl:ObjectProperty rdf:ID=$S:propertyR>
    <owl:inverseOf rdf:resource=$S:propertyP/>
    <$E:inversePropertyElmt
</owl:ObjectProperty>,
<$N:classA rdf:ID=$S:resourceX>
    <$$S:propertyP rdf:resource=$S:resourceY/>
    <$E:XProperties
</$N:classA>,
<$N:classB rdf:ID=$S:resourceY>
    <$E:YProperties
</$N:classB>.```
Derived Information

<Person rdf:about="John">
  <age>29</age>
  <hasChild rdf:resource="#Jill"/>
  <hasAirlineMembership rdf:resource="#tg9000"/>
  <hasParent rdf:resource="#Jack"/>
</Person>

<Person rdf:about="Jill">
  <age>7</age>
  <hasAirlineMembership/>
  <hasParent rdf:resource="#John"/>
</Person>
Language Layers with XDD

XML, XML Schema, NameSpace

Language Layers

Semantic Web Services Ontologies
Domain Ontologies
Ontology Axioms
Application Rules

OWL-S
OWL
RDF, RDF Schema

XML Declarative Description (XDD)

XML, XML Schema, NameSpace

SOAP
Conclusions
Conclusions

- Ontology Engineering (OE) involves the representation, design, development, implementation and application of ontologies
- OE requires an expressive language with efficient computational mechanism
- SWRL = OWL + XMLized subset of Horn logic
- OWL over XDD provides a succinct, expressive OWL+Rules language with efficient computational mechanism