OWL: Introduction to the Web Ontology Language

Jon Atle Gulla

"OWL permits the definition of sophisticated ontologies, a fundamental requirement in the integration of heterogeneous information content."

[Nigel Shadbolt et al., 2004]
OWL Standardization History

- 10 February 2004:
  The World Wide Web Consortium announced final approval of two key Semantic Web technologies
  - The revised Resource Description Framework (RDF)
  - The Web Ontology Language (OWL).

- Motivation:
  - Knowledge representation, and not a message format
  - Availability of tools for reasoning
  - Give information explicit meaning

Semantic Web “Layered Cake”
**Evolution**

- DAML
- OIL
- DAML+OIL
- OWL
- RDF

**OWL Extends RDF**

- OWL uses RDF to define its constructs

```
rdf:Resource
  rdf:Class
  rdf:Property
  owl:Class
  owl:DataType
  owl:ObjectProperty
  owl:DataTypeProperty
```
A Family of Languages

- **OWL Lite** supports users that need primarily a classification hierarchy and simple constraints. It has lower formal complexity than OWL DL.
- **OWL DL** supports users that need maximum expressiveness while retaining computational completeness and decidability.
- **OWL Full** is intended for users who need maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.
  - Class can be treated simultaneously as:
    - a collection of individuals (class extension)
    - and as an individual in its own right (class intention)

- Each of the sublanguages is an extension of its simpler predecessor.

Language Constructs in OWL

- Classes
- Properties
- Property characteristics
- Cardinality
- Individuals
- Others
Classes

- Interpreted as sets of individuals
- Described using formal descriptions that state the requirements for class membership
- Subclasses inherit properties of super classes
- Examples: Person, Car, Town, Fluid, Company

- 6 main ways of describing classes: Named Class; Intersection; Union; Complement; Restrictions; Enumerated classes.

- owl:Class
- rdfs:subClassOf

Union of Classes

- Instances of the Union of two Classes are either the instance of one or both classes. Union Class is formed using union (OR) operator

  Person ≡ Man u Woman

Intersection of Classes

- Instances of the Intersection of two Classes are simultaneously instances of both class. Intersection is formed using AND operator.

  Man ≡ Person n Male

Complement Classes

- Complement class is specified by negating another class. It contains all instances that are not in the negated class

  ¬ Male n Professor
Classes in OWL

- Define Person, man and woman. Man and woman are disjoint classes.

Properties

- Used to state relationships between individuals or from individuals to data values
  
  Eg hasPart, isInhabitedBy, isNextTo, occursBefore

  - **owl:DatatypeProperty** – relations between instances of classes and RDF literals and XML Schema datatypes
  - **owl:ObjectProperty** – relations between instances of two classes
  - **rdfs:subPropertyOf** – hierarchical decomposition of properties
  - **rdfs:domain** - limits the individuals to which the property can be applied
  - **rdfs:range** - limits the individuals that the property may have as its value
Property Characteristics

- **inverseOf** - If the property P1 is stated to be the inverse of the property P2, then if X is related to Y by the P2 property, then Y is related to X by the P1 property.

- **TransitiveProperty** - If a property is transitive, then if the pair (x,y) is an instance of the transitive property P, and the pair (y,z) is an instance of P, then the pair (x,z) is also an instance of P.

- **SymmetricProperty** - If a property is symmetric, then if the pair (x,y) is an instance of the symmetric property P, then the pair (y,x) is also an instance of P.

- **FunctionalProperty** - Properties may be stated to have a unique value.

- **InverseFunctionalProperty** - the inverse of the property has at most one value for each individual. Also referred to as an unambiguous property.

Protege OWL Properties - Interface

- Persons own cars, and cars are owned by persons
How to specify that every car is owned by one person, and that a person can own a maximum of 5 cars?

Cardinality

- \textit{minCardinality} – a way of saying that the property is required to have a value for all instances of the class
- \textit{maxCardinality} - saying that the property is required to have a maximum value for all instances of the class
- \textit{Cardinality} - cardinality is provided as a convenience when it is useful to state that a property on a class has an exact value
Individuals

- Individuals represent objects in the domain
- Individuals are instances of classes and properties
  - may be stated to be equivalent

- Possible to define individuals as:
  - sameAs
  - allDifferent
  - differentFrom

Others

- versionInfo
- backwordCompatibleWith
- priorVersion
- Imports
- incompatibleWith
- DeprecatedClass
- DeprecatedProperty
- allValuesFrom
- someValuesFrom
- equivalentClass
- equivalentProperty
- oneOf
- hasValue
- unionOf
- complementOf
- DataRange
- disjointWith
- intersectionOf
- distinctMembers
- onProperty
- AnnotationProperty
Tool support: Protege

- an ontology editor and a knowledge-base editor
- Free, open-source Java tool that provides an extensible architecture for the creation of customized knowledge-based applications
- developed by the Stanford Medical Informatics group

Protégé-OWL
### Class Hierarchy

- **owl:Thing** is the root class

### Subsumption

- **Superclass/subclass relationship, “isa”**
- **All** members of a subclass can be inferred to be members of its superclasses

- **owl:Thing** is the superclass of all OWL Classes

- **A** subsumes **B**
- **A** is a superclass of **B**
- **B** is a subclass of **A**
- **All** members of **B** are also members of **A**
Disjointness

- OWL assumes that classes overlap

This means an individual could be both a **MeatTopping** and a **VegetableTopping** at the same time
Disjointness

- If it is stated that classes are disjoint

This means an individual cannot be both a MeatTopping and a VegetableTopping at the same time.
- It must be done explicitly in the interface.

ClassesTab: Disjoint Widget

- Add siblings as disjoint
- Remove disjoint siblings
- List of disjoint classes
Properties Tab: Property Browser

Note that Properties can be in a hierarchy

Creating Properties

Delete Property

New Object Property: Associates an individual to another individual

- New Datatype Property (String, int etc)
- New Annotation Properties for metadata
- New SubProperty – ie create “under” the current selection
Conditions

Conditions asserted by the ontology engineer

Add different types of condition

Definition of the class

Description of the class

Conditions inherited from superclasses

Creating Restrictions

Restricted Property

Restriction Type

Filter Expression

Expression Construct Palette

Syntax check
NB! Open World Assumption

hasChild(Jocasta, Oedipus)
hasChild(Jocasta, Polyneikes)
hasChild(Oedipus, Polyneikes)
hasChild(Polyneikes, Thersandros)
Patricide(Oedipus)
¬Patricide(Thersandros)

∃ hasChild.(Patricide □∃ hasChild.¬Patricide))(Jocasta)