Chapter 4
OWL

Based on slides from Grigoris Antoniou and Frank van Harmelen

TL;DR: What is OWL

OWL uses the syntax of RDF but defines new classes and properties, making it **more expressive** as knowledge representation language

Outline

1. **Introduction**
2. Basic Ideas of OWL
3. The OWL Language
4. Examples
5. The OWL Namespace
6. OWL 2

Brief History of OWL

- Builds on RDF to "represent rich and complex knowledge about things, groups of things, and relations between things"
- Draws on decades of experience with systems for representing and reasoning with knowledge
- Based on a 2001 **DAML+OIL** specification
- **OWL** became a W3C recommendation in 2004, extended as **OWL 2** in 2012
- Well defined RDF/XML serializations
- Formal semantics based on first order logic
- Good tools, both opensource and commercial

Outline

1. Introduction
2. **Basic Ideas of OWL**
3. The OWL Language
4. Examples
5. The OWL Namespace
6. OWL 2

Ontology and Data

- Philosophy: Ontologies are models of what exists in the world (kinds of things, relations, events, properties, etc.)
  - Information systems: a schema for info. or data
  - KR languages: model of classes & relations/properties & associated axioms, e.g., subPropertyOf is transitive
- Data is information about individual instances expressed with terms in the ontology
  - Some instances might be considered part of the ontology (e.g., God, George Washington, Baltimore)

Requirements for Ontology Languages

- **Ontology languages** let users write explicit, formal conceptualizations of **domain models**
- Requirements:
  - well-defined syntax
  - efficient reasoning support
  - formal semantics
  - sufficient expressive power
  - convenience of expression

Expressive Power vs. Efficient Reasoning

- Always a tradeoff between expressive power and efficient reasoning support
- The richer the language, the more inefficient the reasoning support becomes (in general)
- Reasoning can be undecidable or semi-decidable and even if decidable can be exponentially hard
- We need a compromise between:
  - Language supported by reasonably efficient reasoners
  - Language that can express large classes of ontologies and knowledge

Kinds of Reasoning about Knowledge

- **Class membership**

  If x is an instance of a class C, and C is a subclass of D, then we can infer that x is an instance of D

- **Equivalence of classes**

  If class A is equivalent to class B, and class B is equivalent to class C, then A is equivalent to C, too

- **Consistency**
  - X is an instance of classes A and B, but A and B are disjoint
  - This is an indication of an error in the ontology or data

- **Classification**

  Certain property-value pairs are a sufficient condition for membership in a class A; if an individual x satisfies such conditions, we conclude that x must be an instance of A

Uses for Reasoning

- **Reasoning support is important for**
  - Deriving new relations and properties
  - Automatically classifying instances in classes
  - Checking consistency of ontology and knowledge
  - checking for unintended relationships between classes
- **Checks like these are valuable for**
  - designing large ontologies, where multiple authors are involved
  - integrating and sharing ontologies from various sources

Reasoning Support for OWL

- Semantics is a prerequisite for reasoning support
- Formal semantics and reasoning support usually provided by
  - mapping an ontology language to known logical formalism
  - using automated reasoners that already exist for those formalisms
- OWL is (partially) mapped to a *description logic*

  DLs are a subset of logic for which efficient reasoning support is possible

RDFS's Expressive Power Limitations

- **Local scope of properties**
  - **rdfs:range** defines range of a property (e.g., eats) for **all** instances of a class
  - In RDF Schema we can't declare range restrictions that apply to only some
  - E.g., animals eat living_things but cows only eat plants
  - :eat rdfs:domain :animal; range :living_thing
    :eat rdfs:domain :cow; range :plant

RDFS's Expressive Power Limitations

- **Disjointness of classes**
  - Sometimes we wish to say that classes are disjoint (e.g., **male** and **female**)
- **Boolean combinations of classes**
  - We may want to define new classes by combining other classes using union, intersection, and complement
  - E.g., **person** equals union of **male** and **female** classes
  - E.g., weekdays equals set {:Monday, … :Sunday}

RDFS's Expressive Power Limitations

- **Cardinality restrictions**
  - E.g., a person has **exactly two** parents, a course is taught by **at least one** lecturer
- **Special characteristics of properties**
  - Transitive property (e.g., *hasAncestor*)
  - Symmetric property (e.g., sibling)
  - Unique property (e.g., *hasMother*)
  - A property is the inverse of another property (e.g., *eats* and *eatenBy*

Combining OWL with RDF Schema

- Ideally, OWL would extend RDF Schema
  - Consistent with the layered architecture of the Semantic Web
- **But** simply extending RDF Schema works against obtaining expressive power and efficient reasoning
  - Combining RDF Schema with logic leads to uncontrollable computational properties
- OWL uses RDF and most of RDFS

Three Species of OWL 1

- W3C’sWeb Ontology Working Group defined OWL as three different sublanguages:
  - OWL Full
  - OWL DL (DL for *Description Logic*)
  - OWL Lite
- Each sublanguage geared toward fulfilling different aspects of requirements

OWL Full

- It uses all the OWL languages primitives
- It allows the combination of these primitives in arbitrary ways with RDF and RDF Schema
- OWL Full is fully upward-compatible with RDF, both syntactically and semantically
- OWL Full is so powerful that its reasoning is undecidable

Soundness and completeness

- A **sound** reasoner only makes conclusions that logically follow from the input, i.e., all of its conclusions are correct
  - We typically require our reasoners to be sound
- A **complete** reasoner can make all conclusions that logically follow from the input
  - We cannot guarantee complete reasoners for full FOL and many subsets
  - So, we can’t do it for OWL Full

OWL DL

- OWL DL (Description Logic) is a sublanguage of OWL Full that restricts application of the constructors from OWL and RDF
  - Application of OWL's constructors to each other is disallowed
  - It corresponds to a well studied description logic
- OWL DL permits efficient reasoning support
- **But** we lose full compatibility with RDF
  - Not every RDF document is a legal OWL DL document
  - Every legal OWL DL document is a legal RDF document

OWL Lite

- An even further restriction limits OWL DL to a subset of the language constructors
  - E.g., OWL Lite excludes enumerated classes, disjointness statements, and arbitrary cardinality
- The advantage of this is a language that is easier to
  - grasp, for users
  - implement, for tool builders
- The disadvantage is restricted expressivity

OWL Compatibility with RDF Schema

- All varieties of OWL use RDF for their syntax
- Instances are declared as in RDF, using RDF descriptions
- OWL constructors are specializations of their RDF counterparts
- OWL classes and properties have additional constraints