OWL: A Description Logic Based Ontology Language

José Luis Ambite
Information Sciences Institute
University of Southern California

based on slides by
Ian Horrocks, University of Manchester
and others

Talk Outline
• Ontologies
• Introduction to Description Logics
• OWL Web Ontology Language
• Protégé OWL Demo
• Resources
• Summary

Ontology (in Computer Science)
• An ontology is an engineering artifact:
  – specific vocabulary used to describe a certain reality
  – explicit assumptions regarding the intended meaning of the vocabulary.
• Thus, an ontology describes a formal specification of a domain:
  – Shared understanding of a domain of interest
  – Formal and machine manipulable model of a domain of interest
• "An explicit specification of a conceptualisation" [Gruber93]

Description Logics
• A family of logic based Knowledge Representation formalisms
  – Descendants of semantic networks
  – Describe domain in terms of concepts (classes), roles (properties, relationships) and individuals
• Distinguished by:
  – Formal semantics (typically model theoretic)
    • Decidable fragments of first-order logic (FOL)
  – Closely related to Propositional Modal & Dynamic Logics
• Inference services:
  – Decision procedures for key problems: satisfiability, subsumption, classification, etc.
  – Implemented systems (highly optimized)

DL Basics
• Concepts: unary predicates/formulae with one free variable p(x)
  – E.g., Person, Doctor, HappyParent, (Doctor u Lawyer)
• Roles: binary predicates/formulae with two free variables r(x,y)
  – E.g., hasChild, loves, (hasBrother o hasDaughter)
• Individuals: constants
  – E.g., John, Mary, Italy
• Concept/Role constructors restricted so that:
  – Satisfiability/subsumption is decidable and, if possible, of low complexity
  – No need for explicit use of variables
    • Realized form of \exists and \forall
  – Features such as counting can be succinctly expressed

The Description Logic Family
• Many description logics: depending on choice of concept/role constructors
• Smallest propositionally closed DL is ALC
  – Concepts constructed using boolean operators:
    • r (and), u (or), ¬ (complement)
  – plus restricted quantifiers
    • ∃ (some), ∀ (all)
  – Only atomic roles
• Example: Person all of whose children are either Doctors or have a child who is a Doctor:
  Person ⊓ ∀hasChild.(Doctor ⊔∃hasChild.Doctor)
The DL Family (2)

- S often used for ALC extended with transitive roles (R+)
- Additional letters indicate other extensions, e.g.:
  - H for role hierarchy (e.g., hasDaughter ⊑ hasChild)
  - O for nominals/singleton classes (e.g., Italy)
  - I for inverse roles (e.g., hasChildOf ≡ hasChild)
  - N for number restrictions (e.g., >2 hasChild, ≤3 hasChild)
  - Q for qualified number restrictions (e.g., >2 hasChild.Doctor)
  - F for functional number restrictions (e.g., ≤2 hasChild)
- S + role hierarchy (H) + inverse (I) + QNR (Q) = SHIQ
- SHIQ is the basis for W3C's OWL Web Ontology Language
  - OWL DL ≈ SHIQ extended with nominals (i.e., SHOIQ)
  - OWL Lite ≈ SHIQ with only functional restrictions (i.e., SHIF)

DL Semantics

- Semantics given by standard FO model theory:
  - Interpretation function I
  - Interpretation domain ∑
  - Individuals i ∈ ∑
  - Concepts C ⊆ ∑
  - Roles r ⊆ ∑ × ∑

- A TBox is a set of "schema" axioms (sentences), e.g.:
  - `{Doctor ⊑ Person, HappyParent ≡ Person ⊓ hasChild(Doctor ⊔ hasChild.Doctor)}`
- An ABox is a set of "data" axioms (ground facts), e.g.:
  - `{John:HappyParent, John hasChild Mary}`
- A Knowledge Base (KB) is just a TBox plus an ABox

The Web Ontology Language OWL

- Semantic Web led to requirement for a "web ontology language"
- W3C set up Web-Ontology (WebOnt) Working Group
  - WebOnt developed OWL language
  - OWL based on earlier languages OIL and DAML+OIL
  - OWL now a W3C recommendation (i.e., a standard)
- OIL, DAML+OIL and OWL based on Description Logics
  - OWL effectively a "Web-friendly" syntax for SHOIN

OWL RDF/XML Exchange Syntax

E.g., Person ⊑ hasChild(Doctor ⊔ hasChild.Doctor):

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType=" collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChildDr"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

OWL Abstract Syntax

E.g., Person ⊑ hasChild(Doctor ⊔ hasChild.Doctor):

```xml
intersectionOf{
  restriction(hasChild allValuesFrom{
    unionOf(Doctor
      restriction(hasChild someValuesFrom(Doctor)))})
}
```
Class/Concept Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>FOL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>( C_1 \cap \ldots \cap C_n )</td>
<td>Human ( \sqcap ) Male</td>
<td>( C_1(x) \land \ldots \land C_n(x) )</td>
</tr>
<tr>
<td>unionOf</td>
<td>( C_1 \cup \ldots \cup C_n )</td>
<td>Doctor ( \sqcup ) Lawyer</td>
<td>( C_1(x) \lor \ldots \lor C_n(x) )</td>
</tr>
<tr>
<td>complementOf</td>
<td>( \neg C )</td>
<td>( \neg \text{Male} )</td>
<td>( \neg C(x) )</td>
</tr>
<tr>
<td>oneOf</td>
<td>{ x_1 \mid \ldots x_n } \land P</td>
<td>(John) ( \land ) (mary)</td>
<td>( x = x_1 \lor \ldots \lor x = x_n \land P[x_1, \ldots, x_n] \rightarrow C(y) )</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>\exists P.C</td>
<td>hasChild.Doctor.relation</td>
<td>( \exists P[x_1, \ldots, x_n] \rightarrow C(y) )</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>\exists P.C</td>
<td>3 hasChild.Lawyer.relation</td>
<td>( \exists P[x_1, \ldots, x_n] \rightarrow C(y) )</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>( \leq n ) P</td>
<td>( \leq 1 ) hasChild</td>
<td>( 3^\leq n ) P(x,y)</td>
</tr>
<tr>
<td>minCardinality</td>
<td>( \geq n ) P</td>
<td>( \geq 2 ) hasChild</td>
<td>( 3^\geq n ) P(x,y)</td>
</tr>
</tbody>
</table>

- \( C \) is a concept (class); \( P \) is a role (property); \( x \) is an individual name
- XMLS datatypes as well as classes in \( \forall P.C \) and \( \exists P.C \)
  - Restricted form of DL concrete domains

Ontology Axioms

<table>
<thead>
<tr>
<th>OWL Syntax</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 ( \sqsubseteq ) Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>( C_1 \equiv C_2 )</td>
<td>Man ( \equiv ) Human ( \equiv ) Male</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>transitiveProperty</td>
<td>( P^+ \sqsubseteq P )</td>
<td>ancestor ( ^+ ) ancestor</td>
</tr>
</tbody>
</table>

- OWL ontology equivalent to DL KB (Tbox + Abox)

Why Description Logic?

- OWL exploits results of 20+ years of DL research
  - Well defined (model theoretic) semantics

Why Description Logic?

- OWL exploits results of 15+ years of DL research
  - Well defined (model theoretic) semantics
  - Formal properties well understood (complexity, decidability)
  - Implemented systems (highly optimised)
Why Description Logic?
- Foundational research was crucial to design of OWL
  - Informed Working Group decisions at every stage, e.g.:
    - “Why not extend the language with feature x, which is clearly harmless?”
    - “Adding x would lead to undecidability - see proof in [...]”

Why Decidable Reasoning?
- OWL’s expressive power restricted so reasoning is decidable
- Design was motivated by:
  - Layered architecture of Semantic Web languages
    - RDF(S) provides basic relational language and simple ontological primitives (or this is what RDF should be)
    - OWL provides powerful but still decidable ontology language
  - Further layers (e.g. RIF) will extend OWL (and may be undecidable)
  - W3C requirement for “implementation experience”
    - Evidence that language can be / is being used “in practice”
    - Should be several implemented systems
  - Users expectations of (automated reasoning) systems
    - Should exhibit correct, consistent and predictable behaviour
    - And should be quick about it!

Tools and Infrastructure
- Editors/environments
  - Protégé, Oiled, Swoop, Construct, Ontotrack, …
- Reasoning systems
  - Cerebra, FaCT++, Kaon2, Pellet, Racer, …

Resources
- Description Logic Handbook, Cambridge University Press
  - http://books.cambridge.org/0521781760.htm
- Description Logic: http://dl.kr.org/
  - complexity: http://www.cs.man.ac.uk/~ezolin/logic/complexity.html
- Web Ontology Language (OWL): http://www.w3.org/2004/OWL/
- Reasoners:
  - FaCT++ (open source): http://owl.man.ac.uk/factplusplus/
  - Racer (commercial): http://www.racer-systems.com/
  - Pellet: http://www.mindswap.org/2003/pellet/
  - (Loom and Powerloom: http://www.isi.edu/isd/LOOM/) 
- Ontology Editors:
  - Protégé: http://protege.stanford.edu/plugins/owl/
- More slides on description logics and OWL:
  - http://www.cs.man.ac.uk/~horrocks/Slides/
Summary

- DLs are a family of logic based KR formalisms
  - Describe domain in terms of concepts, roles and individuals
  - DLs have many applications
- Basis of web ontology language (OWL)