Ontologies and OWL

Riccardo Rosati

Corso di Laurea Magistrale in Ingegneria Informatica
Sapienza Università di Roma
2018/2019
The Semantic Web Tower
Ontology in computer science

• ontology = shared conceptualization of a domain of interest (Gruber, 1993)

• shared vocabulary (set of terms)
  ⇒ simple (shallow) ontology

• (complex) relationships between terms
  ⇒ deep ontology

• AI view:
  – ontology = logical theory (knowledge base)

• DB view:
  – ontology = conceptual data model
Structure of an ontology

- Terms = names for important concepts in the domain
  - Elephant is a concept whose members are a kind of animal
  - Herbivore is a concept whose members are exactly those animals who eat only plants or parts of plants
  - Adult_Elephant is a concept whose members are exactly those elephants whose age is greater than 20 years
- Relationships between terms = background knowledge/constraints on the domain
  - Adult_Elephants weigh at least 2,000 kg
  - All Elephants are either African_Elephants or Indian_Elephants
  - No individual can be both a Herbivore and a Carnivore
Ontology languages

Kinds of potential ontology languages:
- Graphical notations
- Logic-based languages
- Object-oriented languages
- Web schema languages
Ontology languages

- **Graphical notations:**
  - Semantic networks
  - Topic Maps
  - UML
  - RDF
Ontology languages

- **Logic based languages:**
  - Description Logics
  - Rules (e.g., RuleML, Logic Programming/Prolog)
  - First Order Logic (e.g., KIF)
  - Conceptual graphs
  - (Syntactically) higher order logics (e.g., LBase)
  - Non-classical logics (e.g., F-logic, Non-Monotonic Logics, Modal Logics)
Objects/Instanes/Individuals
- Elements of the domain of discourse
- Equivalent to constants in FOL

Types/Classes/Concepts
- Sets of objects sharing certain characteristics
- Equivalent to unary predicates in FOL

Relations/Properties/Roles
- Sets of pairs (tuples) of objects
- Equivalent to binary predicates in FOL
Web schema languages

• Existing Web languages extended to facilitate content description
  – **XML** $\rightarrow$ XML Schema (**XMLS**)  
  – **RDF** $\rightarrow$ RDF Schema (**RDFS**)  

• **XMLS** *not* an ontology language  
  – Changes format of DTDs (document schemas) to be XML  
  – Adds an *extensible type hierarchy*  
    • Integers, Strings, etc.  
    • Can define sub-types, e.g., positive integers  

• **RDFS** *is* recognizable as an ontology language  
  – **Classes** and **properties**  
  – **Sub/super-classes** (and properties)  
  – **Range** and **domain** (of properties)
Limitations of RDFS

- RDFS too weak to describe resources in sufficient detail
  - No localised range and domain constraints
    - Can’t say that the range of hasChild is person when applied to persons and elephant when applied to elephants
  - No existence/cardinality constraints
    - Can’t say that all instances of person have a mother that is also a person, or that persons have exactly 2 parents
  - No transitive, inverse or symmetrical properties
    - Can’t say that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical
  - ...
Web ontology language requirements

Desirable features identified for Web Ontology Language:

- Extends existing Web standards (XML, 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 developed to satisfy above requirements: DAML and OIL

The OWL language (based on DAML+OIL) became a W3C recommendation in 2004
OWL

- OWL = Web Ontology Language
- the OWL family is constituted by 3 different languages (with different expressive power):
  - OWL Full
    - union of OWL syntax and RDF
  - OWL-DL
    - "DL fragment" of OWL Full
  - OWL-Lite
    - "easier to implement" subset of OWL DL
• OWL standards and technology:
  – first version of OWL standardized in 2004
  – reasoning techniques and tools are recent
  – “optimization” of reasoning not fully explored
  – 2009: W3C standardization of OWL 2
## 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>$C_1 \cap \ldots \cap C_n$</td>
<td>Human $\cap$ Male</td>
<td>$C_1 \land \ldots \land C_n$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</td>
<td>Doctor $\cup$ Lawyer</td>
<td>$C_1 \lor \ldots \lor C_n$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \cup \ldots \cup {x_n}$</td>
<td>${\text{john}} \cup {\text{mary}}$</td>
<td>$x_1 \lor \ldots \lor x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall \text{hasChild.Doctor}$</td>
<td>$[P]C$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists \text{hasChild.Lawyer}$</td>
<td>$\langle P\rangle C$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq n P$</td>
<td>$\leq 1 \text{hasChild}$</td>
<td>$[P]_{n+1}$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n P$</td>
<td>$\geq 2 \text{hasChild}$</td>
<td>$\langle P\rangle_n$</td>
</tr>
</tbody>
</table>

- **XMLS datatypes** as well as classes in $\forall P.C$ and $\exists P.C$
  - E.g., $\exists \text{hasAge.nonNegativeInteger}$

- **Arbitrarily complex nesting** of constructors
  - E.g., Person $\cap \forall \text{hasChild.Doctor} \cup \exists \text{hasChild.Doctor}$
RDFS syntax

E.g., concept \( \text{Person} \sqcap \forall \text{hasChild.} \text{Doctor} \sqcup \exists \text{hasChild.} \text{Doctor} \):

\[
\begin{align*}
\text{<owl:Class>} & \\
\text{<owl:intersectionOf rdf:parseType="collection">} & \\
\text{<owl:Class rdf:about="#Person"/>} & \\
\text{<owl:Restriction>} & \\
\text{<owl:onProperty rdf:resource="#hasChild"/>} & \\
\text{<owl:toClass>} & \\
\text{<owl:unionOf rdf:parseType="collection">} & \\
\text{<owl:Class rdf:about="#Doctor"/>} & \\
\text{<owl:Restriction>} & \\
\text{<owl:onProperty rdf:resource="#hasChild"/>} & \\
\text{<owl:hasClass rdf:resource="#Doctor"/>} & \\
\text{<owl:Restriction>} & \\
\text{<owl:unionOf>} & \\
\text{<owl:toClass>} & \\
\text{<owl:Restriction>} & \\
\text{<owl:intersectionOf>} & \\
\text{<owl:Class>}
\end{align*}
\]
## 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>${x_1} \sqsubset \neg {x_2}$</td>
<td>${\text{john}} \sqsubset \neg {\text{peter}}$</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>$T \sqsubseteq \leq 1P$</td>
<td>T $\sqsubseteq$ \leq 1 hasMother</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>$T \sqsubseteq \leq 1P^-$</td>
<td>T $\sqsubseteq$ \leq 1 hasSSN$^-$</td>
</tr>
</tbody>
</table>

Axioms (mostly) reducible to inclusion ($\sqsubseteq$)

\[ C \equiv D \iff \text{both } C \sqsubseteq D \text{ and } D \sqsubseteq C \]
XML Schema datatypes in OWL

- OWL supports **XML Schema** primitive datatypes
  - E.g., integer, real, string, …

- **Strict separation** between “object” classes and datatypes
  - Disjoint interpretation domain $\Delta_D$ for datatypes
    - For a datavalue $d$, $d^I \subseteq \Delta_D$
    - And $\Delta_D \cap \Delta^I = \emptyset$
  - Disjoint “object” and datatype properties
    - For a datatype property $P$, $P^I \subseteq \Delta^I \times \Delta_D$
    - For object property $S$ and datatype property $P$, $S^I \cap P^I = \emptyset$

- Equivalent to the “$(D_n)$” in $SHOIN(D_n)$
OWL DL semantics

• Mapping OWL to equivalent DL ($\mathcal{SHOIN}(D_n)$):
  – Facilitates provision of reasoning services (using DL systems)
  – Provides well defined semantics

• DL semantics defined by interpretations: $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where
  – $\Delta^\mathcal{I}$ is the domain (a non-empty set)
  – $\cdot^\mathcal{I}$ is an interpretation function that maps:
    • Concept (class) name $A \rightarrow$ subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$
    • Role (property) name $R \rightarrow$ binary relation $R^\mathcal{I}$ over $\Delta^\mathcal{I}$
    • Individual name $i \rightarrow i^\mathcal{I}$ element of $\Delta^\mathcal{I}$
OWL DL ontologies are DL knowledge bases

- An OWL ontology maps to a DL Knowledge Base \( \mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle \)
  - \( \mathcal{T}(\text{Tbox}) \) is a set of axioms of the form:
    - \( C \sqsubseteq D \) (concept inclusion)
    - \( C \equiv D \) (concept equivalence)
    - \( R \sqsubseteq S \) (role inclusion)
    - \( R \equiv S \) (role equivalence)
    - \( R^+ \sqsubseteq R \) (role transitivity)
  - \( \mathcal{A}(\text{Abox}) \) is a set of axioms of the form
    - \( x \in D \) (concept instantiation)
    - \( \langle x, y \rangle \in R \) (role instantiation)
OWL vs. RDFS

- **class-def**
- **subclass-of**
- **property-def**
- **subproperty-of**
- **domain**
- **range**

**RDF(S)**

**OWL**

- **class-expressions**
  - AND, OR, NOT
- **role-constraints**
  - has-value, value-type
- **cardinality**
- **role-properties**
  - trans, symm...
OWL vs. First-Order Logic

- in general, DLs correspond to decidable subclasses of first-order logic (FOL)
- DL KB = first-order theory
- OWL Full is NOT a FOL fragment!
  - reasoning in OWL Full is undecidable
- OWL-DL and OWL-Lite are decidable fragments of FOL
let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology about persons where:

- $\mathcal{T}$ contains the following inclusion assertions:
  
  \[
  \text{MALE} \sqsubseteq \text{PERSON} \\
  \text{FEMALE} \sqsubseteq \text{PERSON} \\
  \text{MALE} \sqsubseteq \lnot \text{FEMALE} \\
  \text{PERSON} \sqsubseteq \exists \text{Father}^{-}.\text{MALE}
  \]

- $\mathcal{A}$ contains the following instance assertions:
  
  \[
  \text{MALE} \,(\text{Bob}) \\
  \text{PERSON} \, (\text{Mary}) \\
  \text{PERSON} \,(\text{Paul})
  \]
OWL vs. First-Order Logic

- $\mathcal{T}$ corresponds to the following FOL sentences:
  \[
  \forall x. \text{MALE}(x) \rightarrow \text{PERSON}(x) \\
  \forall x. \text{FEMALE}(x) \rightarrow \text{PERSON}(x) \\
  \forall x. \text{MALE}(x) \rightarrow \neg \text{FEMALE}(x) \\
  \forall x. \text{PERSON}(x) \rightarrow \exists y. \text{Father}(y,x) \text{ and } \text{MALE}(y)
  \]

- $\mathcal{A}$ corresponds to the following FOL ground atoms:
  
  MALE(Bob) \\
  PERSON (Mary) \\
  PERSON(Paul)
Inference tasks in OWL

- Ontology consistency (corresponds to KB consistency in DL)
- Concept/role consistency (same as DL)
- Concept/role subsumption and equivalence (same as DL)
- Instance checking (same as DL)
- …
Inference tasks

• OWL-DL ontology = first-order logical theory
• verifying the formal properties of the ontology corresponds to reasoning over a first-order theory
Inference tasks

- OWL-DL ontology = first-order logical theory
- verifying the formal properties of the ontology corresponds to reasoning over a first-order theory
- main reasoning tasks over ontologies:
  - consistency of the ontology
  - concept (and role) consistency
  - concept (and role) subsumption
  - instance checking
  - instance retrieval
  - query answering
Consistency of the ontology

- Is the ontology $K=(T,A)$ consistent (non-self-contradictory)?
- i.e., is there at least a model for $K$?
- intensional + extensional reasoning task
- fundamental formal property:
  - inconsistent ontology $\Rightarrow$ there is a semantic problem in $K$!
- $K$ must be repaired
Consistency of the ontology

Example TBox:

MALE ⊆ PERSON
FEMALE ⊆ PERSON
MALE ⊆¬ FEMALE
PERSON ⊆∃hasFather.MALE
PERSON ⊆∃hasMother.FEMALE
hasMother ⊆ hasParent
hasFather ⊆ hasParent
∃hasParent.BLACK-EYES ⊆ BLACK-EYES
Consistency of the ontology

Example ABox:

MALE(Bob)
MALE(Paul)
FEMALE(Ann)
hasFather(Ann, Paul)
hasMother(Paul, Mary)
BLACK-EYES(Mary)
¬ BLACK-EYES(Ann)

⇒ TBox + ABox inconsistent (Ann should have black eyes)
Concept consistency

- is a concept definition $C$ consistent in a TBox $T$?
- i.e., is there a model of $T$ in which $C$ has a non-empty extension?
- intensional (schema) reasoning task
- detects a fundamental modeling problem in $T$:
  - if a concept is not consistent, then it can never be populated!
Concept subsumption

- is a concept C subsumed by another concept D in T?
- i.e., is the extension of C contained in the extension of D in every model of T?
- intensional (schema) reasoning task
- allows to do classification of concepts (i.e., to construct the concept ISA hierarchy)
Instance checking

• is an individual $a$ a member of concept $C$ in $K$?
• i.e., is the fact $C(a)$ satisfied by every interpretation of $K$?
• intensional + extensional reasoning task
• basic “instance-level query” (tell me if object $a$ is in class $C$)
Instance retrieval

- find all members of concept C in K
- i.e., compute all individuals a such that C(a) is satisfied by every interpretation of K
- intensional + extensional reasoning task
- (slight) generalization of instance checking
Query answering

- compute the answers to a query $q$ in $K$ (expressed in some query language)
- i.e., compute all tuples of individuals $t$ such that $q(t)$ is entailed by $K$ (= $q(t)$ is satisfied by every interpretation of $K$)
- extensional + extensional reasoning task
- generalization of instance checking and instance retrieval
- e.g.: database queries (SQL-like) over ontologies (or SPARQL-like queries)
Queries over ontologies

classes of queries over DL ontologies considered:

• **conjunctive queries** = subclass of SQL queries
  • correspond to select-project-join queries

• **unions of conjunctive queries**
  • correspond to select-project-join-union queries

• **more expressive queries** (e.g., epistemic queries)

• **SPARQL queries**
  • restrictions/extensions of SPARQL
SPARQL 1.1

- SPARQL 1.1 is the W3C standard query language over OWL ontologies (released in 2013)
- SPARQL 1.1 has different associated entailment regimes that define the semantics of queries over different datasets (RDF models, RDFS+RDF graphs, OWL ontologies)
- the semantics of SPARQL queries for OWL is defined by two entailment regimes for SPARQL:
  - **OWL 2 RDF-based semantics** entailment regime
  - **OWL 2 direct semantics** entailment regime (corresponds to DL semantics)
Computational aspects of reasoning

- reasoning in OWL-DL is decidable (and the complexity is characterized)
- however: high computational complexity (EXPTIME)
- (optimized) reasoning algorithms developed
- OWL-DL reasoning tools implemented
Current OWL technology

two kinds of tools:

• OWL editors ("environments")
• OWL reasoners
OWL editors

- allow for visualizing/browsing/editing OWL ontologies
- able to connect to an external OWL reasoner
  => OWL “environments”
- main current tools:
  - Protege
  - SWOOP
  - OWLed2
OWL reasoning tools

two categories:
• OWL-DL reasoners, e.g.:
  • Hermit
  • Pellet
  • Konclude
  • Racer, RacerPro
  • Fact++
• reasoners for “tractable fragments” of OWL-DL, e.g.:
  • ELK (OWL 2 EL)
  • Mastro, Ontop (OWL 2 QL)
  • RDFox (OWL 2 RL)
OWL-DL reasoning tools

- all tools support “standard” reasoning tasks, i.e.:
  - consistency of the ontology
  - concept consistency
  - concept subsumption and classification
  - instance checking and retrieval
  - query answering (SPARQL)
References

• OWL W3C Web site:
  http://www.w3.org/2004/OWL/

• OWL 2 overview:
  http://www.w3.org/TR/owl2-overview/

• OWL 2 primer:
  http://www.w3.org/TR/2012/REC-owl2-primer-20121211/

• OWL 2 profiles:
  http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/

• OWL 2 quick reference guide:
  https://www.w3.org/TR/owl2-quick-reference/
References

• **SPARQL 1.1:**
  
  http://www.w3.org/TR/sparql11-query/

• **SPARQL 1.1 entailment regimes:**
  
  http://www.w3.org/TR/2013/REC-sparql11-entailment-20130321/

• **Web page on Description Logic reasoners:**
  
  http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/

• **Protege (OWL ontology editor):**
  
  http://protege.stanford.edu/
References

- Hermit (OWL reasoning tool):
  http://hermit-reasoner.com/
- Konclude (OWL 2 DL reasoner):
  http://derivo.de/produkte/konclude/
- ELK (OWL 2 EL ontology reasoner):
  http://www.cs.ox.ac.uk/isg/tools/ELK/
- Stardog (OWL 2 profiles and OWL2 DL reasoner):
  http://stardog.com/
- RacerPro (OWL reasoning tool):
  http://franz.com/agraph/racer/
- Mastro (DL-Lite ontology-based data access system)
  https://www.obdasystems.com/mastro
  https://www.obdasystems.com/mastro-protege-plugin