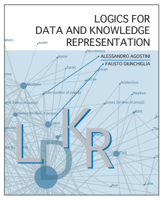
Logics for Data and Knowledge Representation

Alessandro Agostini
agostini@dit.unitn.it

Fausto Giunchiglia
fausto@dit.unitn.it

University of Trento

Ontology Modeling OWL



- Ontologies
- Ontology Languages
  - OWL
- Reasoning
- Appendix: OWL syntax

Ontologies in Computer Science

- Engineering artifact consisting of:
  - A vocabulary used to describe a part of the world (view on a domain of interest).
  - An explicit specification of the intended meaning of the vocabulary.
  - Constrains capturing additional ("meta") knowledge about the depicted domain.

Ontologies in Computer Science

- Ideally, an ontology as an engineering artifact should:
  - capture a shared understanding of a domain of interest;
  - provide a formal and computable (machine manipulable) model (of the domain).

Example (Horrocks *et. al.* 2003)

- A suitable "pizza ontology" might include the information that:
  - Mozzarella and Gorgonzola are kinds of cheese;
  - cheese is not a kind of meat or fish;
  - a vegetarian pizza is one whose toppings do not include any meat or fish.

Example (cont')

- The information (knowledge) provided by the "pizza ontology" allows the term

  "pizza topped with Mozzarella and Gorgonzola"

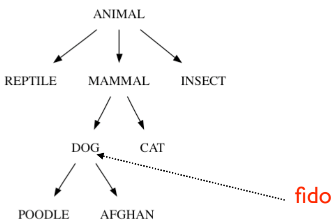
  to be unambiguously interpreted (by, e.g., a pizza ordering agent) as a specialisation of the term "vegetarian pizza".

Ontology (a diagram of)

What is a DL Ontology?

- An ontology is a formal conceptualisation of the "world/domain of interest."
- a DL ontology is a DL **KB** = (TBox,ABox)
- It specifies constraints which declare what should necessarily hold in the world/domain.
- Given an ontology, a legal world description is a possible world satisfying the constraints.

Ontology "Animals" Example



(Etherington & Reiter, 1983)

A DL KB for Animals (Example, cont')

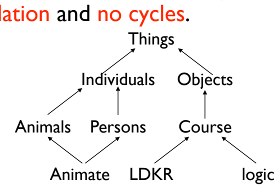
- TBox = { Reptile ⊑ Animal, Mammal ⊑ Animal, Insect ⊑ Animal, Reptile ⊓ Mammal ⊑ ⊥, ...

  Dog ⊑ Mammal, Cat ⊑ Mammal, Dog ⊓ Cat ⊑ ⊥,

  Poodle ⊑ Dog, Afghan ⊑ Dog, Poodle ⊓ Afghan ⊑ ⊥ }
- ABox = { Dog(fido) }.

DL Taxonomy

- A DL taxonomy is a terminology (i.e. set of DL concept names) partially ordered by a subsumption relation and no cycles.
- Example:



Where are Ontologies used?

- e-Science: bioinformatics, ...
- Medicine
- Databases: schema design, sharing, integration, matching, query-answering, ...
- User Interfaces
- Semantic Web/Grid
- Library Science: (subject) classification, ...

Examples

- E-commerce: ontologies facilitate communication between buying and selling agents by providing a common vocabulary to describe goods (such as pizzas) and services.
- Search engines: ontologies help in finding pages that contain semantically similar but syntactically different words and phrases.

Web Catalogs

- Web catalogs (e.g., Yahoo! dmoz), use structured vocabularies (i.e., taxonomies), for indexing the objects of a domain:
  - less pages indexed w.r.t. search engines using statistical methods (e.g., AltaVista)...
  - ...but higher classification quality as it is hand-crafted by domain experts.

Web Catalogs (cont')

- Web catalog are *almost always* taxonomies:
  - The nodes correspond to terms (e.g. Sciences, Mathematics) and the edges correspond to subsumption relationships (e.g., Mathematics $\sqsubseteq$ Sciences).
- Such taxonomies may contain thousands of terms (e.g. Yahoo! contains 20K terms; Dmoz contains 300K terms...)

Ontologies: History

- A philosophical (metaphisic) discipline aimed to understand and organize the reality and the human ("Science of Being" Aristotele).
- Importantly, ontologies historically used to organizing knowledge in a domain of interest
  - Classification (e.g. Library science, CC, DDC, UDC Systems, ...)

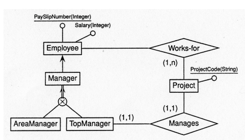
Photo "Bodleian Library - Oxford" by Chris Donaghue, © The Oxford Photo Library - CP0497

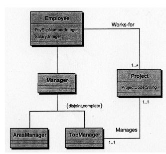
Ontology Languages

- Ontology languages are typically expressed:
  - by means of diagrams (graphs), such as
  - Semantic Networks
  - UML
  - RDF (Resource Description Framework)
  - RDFS (RDF Schema)
- - by logic (FOL, DLs), such as OWL.

Ontology Languages (Example)

- The Entity-Relationship Diagrams and the UML Class Diagrams can be considered as graph-based ontology languages.





Requirements for OLs

- OLs allow users to write explicit, formal conceptualizations of domain models.
- The main requirements are:

  - a well-defined syntax
  - a formal semantics
  - an efficient reasoning support
  - sufficient expressive power
  - convenience of expression

Expressivity and Computability

- The richer the ontology language is, the more inefficient the reasoning services become.
- Some OL may have noncomputable services!
- We need a compromise:
  - an OL with efficient reasoning services;
  - an OL that can express concepts an knowledge viz ontologies we need to.

“Schema” Languages (1)

- Existing ontology (web) languages extended to facilitate content description:
  - XML ⇒ XML Schema (XMLS)
  - RDF ⇒ RDF Schema (RDFS)
- XMLS *is not* an ontology language
- RDFS *is* an ontology language
- ... see next

“Schema” Languages (2)

- XMLS *is not* an ontology language:
  - changes format of DTDs to be XML
  - adds an extensible type hierarchy:
    * integers, stringes
    * subtypes (e.g. positive integers)
- RDFS *is* an ontology language:
  - classes and properties
  - sub/super classes and properties
  - range and domain of properties

Limitations of RDFS (Expressive Power)

- The RDFS ontology language has some strong limitations in its expressive power:
  - Local scope of properties
  - Disjointness of classes
  - Boolean combinations of classes
  - Cardinality restrictions
  - Special characteristic of properties

Limitations of RDFS (1)

- **Local scope of properties:**
  - **rdfs:range** defines the range of a property for all classes.
  - Example: take property (concept) ‘read’; in RDFS we cannot say that ‘read’ applies to books, newspapers, or magazines.
- Thus, RDFS cannot express a property’s restrictions that apply only to some classes.

Limitations of RDFS (2)

- **Disjointness of classes:**
  - In RDFS we cannot express disjoint classes or partitions, e.g. Meat and Cheese.
- **Boolean combinations of classes:**
  - In RDFS we cannot define new classes as boolean combinations of existing classes.
  - Example: Body as union of Arms and Head.

Limitations of RDFS (3)

- **Cardinality restrictions:**
  - In RDFS we cannot express restrictions on the number of objects a property applies.
  - Example 1: Mammal has at-most 4 Legs.
  - Example 2: Person has exactly 2 Parents.
  - ...

Limitations of RDFS (4)

- **Special characteristic of properties:**
  - In RDFS we cannot define many important properties, we mention:
    - transitivity - e.g. "is greater than"
    - functionals - e.g. "is mother of"
    - inverse - e.g. "hasChild" for "isChildOf"
    - symmetrical - e.g. "touches"

From RDF to OWL

- OWL is defined as an extension to RDF in the form of a vocabulary entailment:
  - the syntax of OWL is the syntax of RDF;
  - the semantics of OWL is an extension of the semantics of RDF/RDFS.
  - OWL uses RDF's XML-based syntax.
  - Alternative syntax: abstract, UML-based,...

OWL RDF/XML Exchange Syntax

```
<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="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

(Horrocks CISA-06)

OWL and DLs

- OWL RDF/XML syntaxt is verbose, much more than DL syntax!

```
<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="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

How do we express it in DL?

OWL and DLs

- Person $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
- **Exercise:** How you represent it in DL?

```
<owl:Class rdf:ID="Student">
   <owl:intersectionOf rdf:parsetype="Collection">
     <owl:Class rdfs:about="Person" />
     <owl:Restriction>
        <owl:onProperty rdf:resource="enrolledIn" />
        <owl:minCardinality rdfs:datatype="&xsd;Integer">
                1
        </owl:minCardinality>
     </owl:Restriction>
   <owl:intersectionOf>
</owl:Class>
```

(Horrocks *et. al.* JWS-03)

OWL - Introduction

- In order to allow sharing and reuse of ontologies on the Semantic Web, a <u>common ontology language is required</u>.
- The W3C has developed two ontology languages for use on the Semantic Web:
  - RDFS [Brickley & Guha, 2004], developed as a lightweight ontology language.
  - OWL [Dean & Schreiber, 2004] ...see next

OWL - Introduction

- OWL [Dean & Schreiber, 2004] is a <u>more expressive ontology language</u> based on DLs.
- Developed by W3C's Web-Ontology (WebOnt) Working Group (2004).
- Starting language was an extension of RDF/RDFS languages, called DAML+OIL.
- DAML+OIL is a combination of American language DAML-ONT and European's OIL.

OWL - Introduction

- Now a W3C Recommendation (i.e. a standard, like HTML and XML).
- OWL (and DAML+OIL as well) is based on description logic, in particular *SHOIN*(**D**) DL.
- In fact OWL is a "web friendly" syntaxt for *SHOIN*(**D**) (quote attributed to I. Horrocks).

Three Species of OWL

- OWL consists of three "species," namely OWL Lite, OWL DL and OWL Full.
- These languages are intended to be layered according to increasing expressiveness.
- Each language is based on a specific description logic (we see it in a few slides).
- OWL Lite / OWL DL by far the most used.

OWL Full, DL, Lite

- OWL Full : union of OWL syntax and RDF.
  - RDF semantics extended with relevant semantic conditions and axiomatic triples.
- OWL DL : restricted to DL/FOL fragment (DAML+OIL).
  - We are mostly interested in OWL DL.
- OWL Lite : subset of OWL DL easier to implement; tools/implemantations available.

OWL: Lite versus DL

- It turns out that OWL DL adds very little in expressiveness to OWL Lite [Horrocks & Patel-Schneider, JWS 2003].
- OWL Lite and OWL DL pose several restrictions on the use of RDF and redefine the semantics of the RDFS primitives.
- **NB:** OWL Lite and OWL DL are *not properly layered* on top of RDFS.

OWL: DL versus Full

- OWL Full *layers on top of* both RDFS and OWL DL.
- **NB:** Because of RDFS and OWL DL are so different, the semantics of OWL Full is not a proper extension of OWL DL's semantics:

  In fact, OWL DL has a model-theoretic semantics;RDFS has an axiomatic semantics. RDFS has also a more syntactical freedom.

Layering Problems

- The lack of proper layering between
  (1) RDFS and OWL DL / OWL Lite, and
  (2) OWL DL / OWL Lite and OWL Full
  raises doubts about interoperability between ontologies written in these languages.
- Problems arise especially in the areas of Software Engineering and Database Systems.

  [Bruijn, Pollers, Lara & Fensel, 2005]

OWL as a DL-based Language

- Without regarding annotation properties of OWL as a web language (cf. RDF/XML),
  - OWL Lite is equivalent to *SHIF*(**D**) DL;
  - OWL DL is equivalent to *SHOIN*(**D**) DL.
- So, an OWL ontology is equivalent to a DL knowledge base (TBox + ABox).

*S**-family of DLs

- More expressive description logics (DLs) are usually extensions of some *AL**-based DL.
- **Starting point**: *S* denotes the *ALC*-based DL with transitive role axioms (Trans(R)). Thus *S*'s Tboxes extends *ALC*'s Tboxes with transitive role axioms.
- "S" stands for 'Subsumption,' as a reminder that this logic models concept axioms C⊑D.

*S**-family of DLs

- Each DL L that extends *S* is denoted by a string *S*[*I*][*H*][*N*][*Q*][*F*][*O*], where:
  I : L allows Inverse roles
  H : L allows role inclusion axioms (i.e. role Hierarchies as finite sets of role axioms)
  N : L allows Number restrictions
  Q : L allows Qualified number restrictions
  F : L allows Functional number restrictions
  O : L allows nOminal/singleton classes.

*S**-family of DLs Examples

- *I* : DL allows *I*nverse roles:
  E.g.: hasChild ≡ isChildOf⁻
- *H* : DL allows role inclusion axioms:
  E.g.: isDirectPartOf ⊑ isPartOf
- *N* : DL allows *N*umber restrictions:
  E.g.: ≥2hasArm ⊓ ≤2hasArm (ie =2hasArm)
- *Q* : DL allows *Q*ualified number restrictions:
  E.g.: ≥2hasArm.Body, ≥2hasArm.⊤.

*S**-family of DLs Examples (cont')

- *F* : DL allows *F*unctional number restrictions:
  E.g.: ≤1hasMother (i.e. ≤nR for n=1)
- *O* : DL allows n*O*minal classes, i.e. to define a class by enumerating its instances.
  E.g.: {Trentino}, {Mon,Wed,Fri,Sun}.
- **Remark:** Since *ALF*⊆*ALN*⊆*ALQ*, any logic of the form *S*Q* extends any logic of the form *S*N or S*F*. In particular, *SHIQ* extends *SHIN*.

From *SHIQ* to OWL

- *S* + role hierarchy (*H*) + inverse roles (*I*) + qualified number restrictions (*Q*) = *SHIQ*
- *SHIQ* is the basis for W3C's OWL Web Ontology Language
  - OWL DL : *SHIQ* extended with nominals and datatypes, *N* for *Q* (i.e. *SHOIN*(**D**)).
  - OWL Lite : *SHIQ* extended with functionals and datatypes, no *Q* (*SHIF*(**D**)).

Datatypes

- (Concrete) datatypes are used to represent literal values such as numbers and strings.
- A type system typically defines a set of "primitive" datatypes, such as *string* or *integer*, and provides a mechanism for deriving new datatypes from existing ones.

Reasoning Services on Ontology Knowledge

- Premise: see also reasoning services in DLs.
- **Class membership** (Instance checking): If individual *a* is an instance of a class C and C is a subclass of D, then infer that *a* is an instance of C.
- **Equivalence of classes**: If class A is equivalent to class B and B is equivalent to class C, then A and C are equivalent.

Reasoning Services on Ontology Knowledge

- **Consistency**: An individual *a* is an instance of classes A and B, but A and B are disjoint.
  - Consistency checking allows to discover an error in the ontology.
- **Classification**: Certain property-value pairs are a sufficient condition for membership in a class (cf. definitions in DLs). If *a* satisfies such conditions we classified it!

Uses of Reasoning Services (1)

- Reasoning services are important for:
  - (automatically) checking the consistency of an ontology and the knowledge therein;
  - (automatically) checking for unintended relationships between classes;
  - (automatically) classifying "objects" from a domain of interest (Δ, the "world") into classes (concepts).

Uses of Reasoning Services (2)

- Checking like the preceding are used for:
  - design and maintain high quality / large / complex / distributed ontologies
  - integrating, sharing, matching of ontologies from different sources
  - correct and capture intuitions of experts,
  - answer queries, retrieve objects/tuples, ...

Reasoning Services for OWL

- Formal semantics is a prerequisite for reasoning services (please see slides on DLs).
- Semantics and reasoning services are usually provided by:
  - Semantics: mapping the ontology language to a known formalism - i.e. a certain DL!
  - Reasoning: using some automated reasoners existing for that formalism.

OWL Reasoning

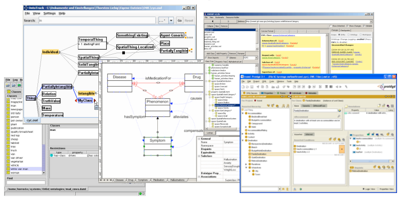
- Computing ontology entailment in OWL DL (OWL Lite) has the same complexity as computing KB SAT in *SHOIN*(**D**) (*SHIF*(**D**)).
- DL algorithms and implementations can be used to provide reasoning services for OWL Lite. [Horrocks & Patel-Schneider, ISWC-03]
- The design of "practical" algorithms for *SHOIN*(**D**) is still a hot research topic.

Using Standard DL Techniques

- State of the art DL systems typically use highly optimised tableaux algorithms to decide KB satisfiability (consistency).
- Tableaux algorithms work by trying to construct a concrete example (i.e. a model) consistent with the KB axioms. Two steps:
  - start from Abox axioms (the ground facts)
  - check implications using TBox axioms.

Tools and Infrastructure

- Editors and environments:
  Olied, Protege, Swoop, Construct, etc.



Tools and Infrastructure

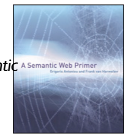
- Reasoning Systems:
  Cerebrea, FaCT++, Pellet, Racer, Kaon2, ...

Cerebra FaCT++
Racer Pellet

Some Resources

- FaCT++ system (open source):
  - http://owl.man.ac.uk/factplusplus/
- Protege system:
  - http://protege.stanford.edu/plugins/owl/
- W3C:
  - http://www.w3.org/2004/OWL/
- DL Handbook:
  - http://books.cambridge.org/0521781760.html

Some Resources



- Books:
  - G. Antoniou and F. van Harmelen, *A Semantic Primer*. The MIT Press, 2004. (Chs 1, 4)
    http://www.ics.forth.gr/isl/swprimer/
- Papers & Links (if any):
  - http://dit.unitn.it/~ldkr/#Biblio