

Ontologies: An Introduction

Claudia d'Amato

Co-Author: Nicola Fanizzi

Dipartimento di Informatica • Università degli Studi di Bari
Campus Universitario, Via Orabona 4, 70125 Bari, Italy

Bari, November, 7 2007

Contents

- 1 Introduction
 - Origin of Ontology
 - Ontology: Current Meanings
 - Kinds of Ontologies
 - Ontology Usage
- 2 Writing an Ontology
 - Ontology: Basics Elements
 - The Reference Representation Language
 - Knowledge Base Definition
 - Reasoning Services
 - OWL Expressiveness
 - OWA vs. CWA
- 3 Ontology Tools
- 4 Conclusions

The Origin of the "Ontology"

The term **Ontology** has its origin in philosophy

- **Ontology: the branch of philosophy which deals with the nature and the organisation of reality**

In this sense the Ontology tries to answer to the question:

- *What is being?* or, in a meaningful reformulation:
- *What are the features common to all beings?*

"Ontology": recently adopted in several fields of computer science and information science \Rightarrow several meaning have been assigned to the "Ontology" term

"Ontology" in Computer Science and Information Science

[Guarino et al. 1995]

- Ontology as a specific "*syntactic*" object
 - Ontology as representation of conceptual system via a logical theory
 - Ontology as the vocabulary used by a logical theory
 - Ontology as a meta-level specification of a logical theory
- Ontology as a conceptual "*semantic*" entity
 - Ontology as *informal* conceptual system
 - Ontology as *formal* semantic account
- Ontology as specification of a *conceptualization*
 - an intensional semantic structure which encodes the implicit rules constraining the structure of a piece of reality.

Different kinds of Ontologies

- **Domain Ontology:** models a specific domain. It represents the particular meanings of terms as they apply to that domain (ex. *CHEMICALS*, *Gene Ontology*). The word *card* has different meaning:
 - An ontology about the domain of *poker* would model the *playing card*
 - An ontology about the domain of *computer hardware* would model the *punch card* and *video card* meanings.
- **Upper Ontology** (or foundation ontology): is a model of the common objects that are generally applicable across a wide range of domain ontologies.
 - It contains a core glossary in whose terms can be used to describe a set of domains. Ex. *Dublin Core*, *GFO*, *OpenCyc/ResearchCyc*, *SUMO*, and *DOLCE*

Ontology Role and Usage

Def. An *ontology* is a formal conceptualization of a domain that is shared and reused across domains, tasks and group of people [A. Gomez Perez et al. 1999]

- The *ontology role* is to make *semantics explicit*. Thus ontologies are *used* for:
 - Constituting a community reference
 - Sharing consistent understanding of what information means
 - Making possible Knowledge Reuse and Sharing
 - Increasing Interoperability between systems

Ontology: Basic elements...

- **Individuals:** are the "ground level" components of an ontology.
 - Individuals can be: **1) concrete objects of a domain** i.e. *people, animals, automobiles, molecules...* **2) abstract individuals** i.e. *numbers and words*.
- **Concepts:** are collections of objects. They may contain individuals, other classes, or a combination of both. Some examples of classes:
 - *Molecule*, the class of all molecules
 - *Vehicle*, the class of all vehicles
 - *Car*, the class of all cars

...Ontology: Basic elements

- **Attributes:** *describe the objects* in the ontology.
Ex.: the Ford Explorer object has attributes:
 - Number-of-doors: 4
 - Transmission: 6-speed
- **Relationships:** make explicit the links between objects.
A relationship can be model as:
 - ① an *attribute whose value is another object* in the ontology,
 - Ex.: given the objects Ford Explorer and Ford Bronco, the attribute Successor:Ford Explorer of Ford Bronco means that Explorer is the modeled that replaced Bronco.
 - ② a mathematical relation
 - Ex.: Successor(Ford Bronco,Ford Explorer)
- Much of the power of ontologies comes from the ability to describe these relations.

Ontology Representation Languages

- **Informal:** natural language
- **Semi-Formal:** limited structured form of natural language
- **Formal:** formal language with formal semantics
 - *Cycl*: developed in the Cyc project. It is based on first-order predicate calculus with some higher-order extensions.
 - *RIF* (Rule Interchange Format) and *F-Logic*: combine ontologies and rules.
 - **OWL**: developed as a follow-on from RDF and RDFS, and earlier ontology language projects: OIL, DAML, DAML+OIL. *OWL is intended to be used over the World Wide Web*,
 - Supported by *well-founded semantics* of *DLs*
 - together with a series of available automated *reasoning services* allowing to derive logical consequences from an ontology

DL: The Reference Representation Language

Basics elements of DL

- Primitive *concepts* $N_C = \{C, D, \dots\}$: subsets of a domain
- Primitive *roles* $N_R = \{R, S, \dots\}$: binary relations on the domain
- *Interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$: *domain* of the interpretation and $\cdot^{\mathcal{I}}$: *interpretation function* that assigns to each primitive concept C a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and assigns to each primitive role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Building Complex Concept Descriptions

Name	Syntax	Semantics
atomic negation	$\neg A, A \in N_C$	$A^I \subseteq \Delta^I$
full negation	$\neg C$	$C^I \subseteq \Delta^I$
concept conj.	$C \sqcap D$	$C^I \cap D^I$
concept disj.	$C \sqcup D$	$C^I \cup D^I$
full exist. restr.	$\exists R.C$	$\{a \in \Delta^I \mid \exists b (a, b) \in R^I \wedge b \in C^I\}$
universal restr.	$\forall R.C$	$\{a \in \Delta^I \mid \forall b (a, b) \in R^I \rightarrow b \in C^I\}$
at most restr.	$\leq nR$	$\{a \in \Delta^I \mid \{b \in \Delta^I \mid (a, b) \in R^I\} \leq n\}$
at least restr.	$\geq nR$	$\{a \in \Delta^I \mid \{b \in \Delta^I \mid (a, b) \in R^I\} \geq n\}$
qualif. at most r.	$\leq nR.C$	$\{a \in \Delta^I \mid \{b \in \Delta^I \mid (a, b) \in R^I \wedge b \in C^I\} \leq n\}$
qualif. at least r.	$\geq nR.C$	$\{a \in \Delta^I \mid \{b \in \Delta^I \mid (a, b) \in R^I \wedge b \in C^I\} \geq n\}$
one-of	$\{a_1, a_2, \dots, a_n\}$	$\{a \in \Delta^I \mid a = a_i, 1 \leq i \leq n\}$
has value	$\exists R.\{a\}$	$\{b \in \Delta^I \mid (b, a^I) \in R^I\}$
inverse of	R^-	$\{(a, b) \in \Delta^I \times \Delta^I \mid (b, a) \in R^I\}$

Knowledge Base & Subsumption

$$\mathcal{K} = \langle \mathcal{I}, \mathcal{A} \rangle$$

- *T-box* \mathcal{T} is a set of definitions $C \equiv D$, meaning $C^{\mathcal{I}} = D^{\mathcal{I}}$, where C is the concept name and D is a description
- *A-box* \mathcal{A} contains extensional assertions on concepts and roles e.g. $C(a)$ and $R(a, b)$, meaning, resp., that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Subsumption

Given two concept descriptions C and D , C *subsumes* D , denoted by $C \sqsupseteq D$, iff for every interpretation \mathcal{I} , it holds that $C^{\mathcal{I}} \supseteq D^{\mathcal{I}}$

TBox: Example

Primitive Concepts: $N_C = \{\text{Female, Male, Human}\}$.

Primitive Roles:

$N_R = \{\text{HasChild, HasParent, HasGrandParent, HasUncle}\}$.

$\mathcal{T} = \{ \text{Woman} \equiv \text{Human} \sqcap \text{Female}; \text{Man} \equiv \text{Human} \sqcap \text{Male}$

$\text{Parent} \equiv \text{Human} \sqcap \exists \text{HasChild.Human}$

$\text{Mother} \equiv \text{Woman} \sqcap \text{Parent}$

$\text{Father} \equiv \text{Man} \sqcap \text{Parent}$

$\text{Child} \equiv \text{Human} \sqcap \exists \text{HasParent.Parent}$

$\text{Grandparent} \equiv \text{Parent} \sqcap \exists \text{HasChild} . (\exists \text{HasChild.Human})$

$\text{Sibling} \equiv \text{Child} \sqcap \exists \text{HasParent} . (\exists \text{HasChild} \geq 2)$

$\text{Niece} \equiv \text{Human} \sqcap \exists \text{HasGrandParent.Parent} \sqcup \exists \text{HasUncle.Uncle}$

$\text{Cousin} \equiv \text{Niece} \sqcap \exists \text{HasUncle} . (\exists \text{HasChild.Human}) \}$.

ABox: Example

$\mathcal{A} = \{ \text{Woman}(\text{Claudia}), \text{Woman}(\text{Tiziana}), \text{Father}(\text{Leonardo}), \text{Father}(\text{Antonio}),$
 $\text{Father}(\text{AntonioB}), \text{Mother}(\text{Maria}), \text{Mother}(\text{Giovanna}), \text{Child}(\text{Valentina}),$
 $\text{Sibling}(\text{Martina}), \text{Sibling}(\text{Vito}), \text{HasParent}(\text{Claudia}, \text{Giovanna}),$
 $\text{HasParent}(\text{Leonardo}, \text{AntonioB}), \text{HasParent}(\text{Martina}, \text{Maria}),$
 $\text{HasParent}(\text{Giovanna}, \text{Antonio}), \text{HasParent}(\text{Vito}, \text{AntonioB}),$
 $\text{HasParent}(\text{Tiziana}, \text{Giovanna}), \text{HasParent}(\text{Tiziana}, \text{Leonardo}),$
 $\text{HasParent}(\text{Valentina}, \text{Maria}), \text{HasParent}(\text{Maria}, \text{Antonio}), \text{HasSibling}(\text{Leonardo}, \text{Vito}),$
 $\text{HasSibling}(\text{Martina}, \text{Valentina}), \text{HasSibling}(\text{Giovanna}, \text{Maria}),$
 $\text{HasSibling}(\text{Vito}, \text{Leonardo}), \text{HasSibling}(\text{Tiziana}, \text{Claudia}),$
 $\text{HasSibling}(\text{Valentina}, \text{Martina}), \text{HasChild}(\text{Leonardo}, \text{Tiziana}),$
 $\text{HasChild}(\text{Antonio}, \text{Giovanna}), \text{HasChild}(\text{Antonio}, \text{Maria}), \text{HasChild}(\text{Giovanna}, \text{Tiziana}),$
 $\text{HasChild}(\text{Giovanna}, \text{Claudia}), \text{HasChild}(\text{AntonioB}, \text{Leonardo}),$
 $\text{HasChild}(\text{Maria}, \text{Valentina}), \text{HasUncle}(\text{Martina}, \text{Giovanna}),$
 $\text{HasUncle}(\text{Valentina}, \text{Giovanna}) \}$

From DL to OWL

```
<owl:Class rdf:ID="Human" />
<owl:Class rdf:ID="Father" >
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection" >
        <owl:Class rdf:ID="Man" />
        <owl:Class rdf:ID="Parent" />
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="Female" >
  <owl:disjointWith>
    <owl:Class rdf:ID="Male" />
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="Child" >
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:someValuesFrom>
```

Reasoning on ontology

- An ontology is made by a set of axioms \Rightarrow *implicit knowledge* can be *made explicit* (derived) through inferences.
 - Developed Reasoners (based on DL formal semantics):
 - *FaCT*
 - *RACER*
 - *PELLET*

TBox Standard Inference Services

- *Concept Satisfiability*: checks if a newly defined concept makes sense w.r.t. the existing TBox or it is contradictory
 - **Ex.:** $\mathcal{T} = \{ \text{Parent, Man, Woman} \equiv \neg \text{Man}, \text{Mother} \equiv \text{Woman} \sqcap \text{Parent} \}$ *added* $\text{Man} \sqsupseteq \text{Mother} \Rightarrow$ the new axiom is *unsatisfiable* w.r.t TBox since the disjointness constraint between Man and Woman is violated
- *Subsption*: checks if a concept C is more general than another concept $D \Rightarrow$ used for computing *concept hierarchy*
 - **Ex.:** $\mathcal{T} = \{ \text{Parent, Man, Woman} \equiv \neg \text{Man}, \text{Mother} \equiv \text{Woman} \sqcap \text{Parent} \}$ *added* $\text{Father} \equiv \text{Man} \sqcap \text{Parent} \Rightarrow$ $\text{Parent} \sqsupseteq \text{Father}$ and $\text{Man} \sqsupseteq \text{Father}$

ABox Standard Inference Services

- *ABox consistency (w.r.t. the TBox)*: checks if a new (concept or role) assertion in an ABox \mathcal{A} makes \mathcal{A} inconsistent w.r.t. a TBox \mathcal{T} or not
 - **Ex.1:** $\mathcal{T} = \{\text{Woman} \equiv \text{Person} \sqcap \text{Female}, \text{Man} \equiv \text{Person} \sqcap \neg \text{Female}\}$; $\mathcal{A} = \{\text{Woman}(\text{MARY}), \text{Man}(\text{MARY})\} \Rightarrow \mathcal{A}$ is *inconsistent* w.r.t. \mathcal{T}
 - **Ex.2:** TBox $\mathcal{T} = \{\text{Woman}, \text{Man}\} \Rightarrow \mathcal{A}$ is *consistent* w.r.t. \mathcal{T} since no restrictions are imposed on the interpretation of Woman and Man
- *Instance Checking*: decide whether an individual is an instance of a concept or not
- *Retrieval*: finds all individuals instance of a concept
 - **Ex.:** $\mathcal{T} = \{\text{Female} \sqsupseteq \text{Woman}\}$; $\mathcal{A} = \{\text{Female}(\text{Ann}), \text{Woman}(\text{Sara})\} \Rightarrow \text{Retrieval}(\text{Female}) = \{\text{Ann}, \text{Sara}\}$

Reasoning Services in Ontology Life-Cycle

Reasoning services can be employed in different phases of the ontology life-cycle

- **Ontology design**

- Check concept satisfiability, ontology satisfiability and (unexpected) implied relationships

- **Ontology aligning and merging**

- Assert inter-ontology relationships
- Reasoner computes integrated concept hierarchy/consistency

- **Ontology deployment**

- Determine if a set of facts are consistent w.r.t. ontology
- Determine if individuals are instances of ontology concepts
- Classification-based querying

Expressiveness and Computability

With the increasing of the expressive power of the knowledge representation language the computational complexity of the reasoning procedure increases \Rightarrow increasing of the time necessary for computing inferences

- It can happen that some inferences do not give any reply \Rightarrow semi-decidable procedure

If a conclusion C is not a logic consequence of a set of premises P then the procedure for its prove cannot terminate.

OWL Languages

The OWL language provides three increasingly expressive sublanguages:

- **OWL Lite:** *decidable* with desirable computational properties
 - supports the classification hierarchy inference and simple constraint features, i.e. cardinality constraints on properties where only cardinality values of 0 and 1 are permitted.
- **OWL DL:** *decidable* but subject to higher worst-case complexity. It is so called for its correspondence with DL.
 - allows restrictions such as type separation (a class cannot also be an individual or a property, a property cannot also be an individual or a class).
- **OWL Full:** *not decidable*
 - a class can be treated simultaneously as a collection of individuals and as an individual in its own right.

OWA vs. CWA

- **Open World Assumption:** typical of DL
 - Absence of information is interpreted as unknown information
- **Closed World Assumption:** typical of DB
 - Absence of information is interpreted as negative information

Ex.: $\mathcal{T} = \{ \text{Female}, \text{Woman} \};$
 $\mathcal{A} = \{ \text{Female}(\text{Ann}), \text{Woman}(\text{Sara}) \}$

CWA: $q = \text{Female}(\text{Sara}) ? \rightarrow \text{NO}$

OWA: $q = \text{Female}(\text{Sara}) ? \rightarrow \text{UNKNOWN}$

Tools for building and managing ontologies

- **Protégé:** free, open source ontology editor and knowledge-base framework
- **Chimaera:** system for creating and maintaining distributed ontologies on the web. Major supported functions: merging multiple ontologies; diagnosing of one or more ontologies.
- **Ontolingua:** distributed collaborative environment to browse, create, edit, modify, and use ontologies.
- **OntoEdit:** Engineering Environment for the development and maintenance of ontologies using graphical means.
- **WebOnto:** Java applet coupled with a customised web server allowing to browse and edit knowledge models over the web.
- **KAON:** open-source infrastructure for ontology creation and management, and providing a framework for building ontology-based applications.

Conclusions

Summarizing

- An *ontology* is a formal conceptualization of a domain that is shared and reused across domains, tasks and group of people
- Ontologies are *used* in different fields *for*:
 - Constituting a community reference
 - Sharing consistent understanding of what information means
 - Making possible interoperability between systems
 - Making the *Web* machine-readable and processable besides of human-readable (Semantic Web)

Line of research:

- *Ontology construction* is the *result of a complex process* of knowledge acquisition \Rightarrow (semi)-automatic tools for building ontologies are necessary
 - *Machine learning methods* can be useful for accomplishing such a goal

The End

That's all!

Claudia d'Amato

`claudia.damato@di.uniba.it`

Nicola Fanizzi

`fanizzi@di.uniba.it`

Bibliography

- N. Guarino and P. Giaretta. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In Proc. of 2nd Int. Conf. on Building and Sharing Very Large-Scale Knowledge Bases (1995).
- A. Gomez Perez and V.R. Benjamins. Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. In Proc. of Ontology and Problem-Solving Methods: Lesson learned and Future Trends, Workshop at IJCAI, vol. 18, pages 1.11.15. CEUR Publications, Amsterdam, 1999.
- [*http : //en.wikipedia.org/wiki/Ontology_\(computer_science\)*](http://en.wikipedia.org/wiki/Ontology_(computer_science))
- [*http : //www.w3.org/2004/OWL/*](http://www.w3.org/2004/OWL/)

Bibliography

- John F. Sowa, Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA, 2000
[http : //www.jfsowa.com/krbook/](http://www.jfsowa.com/krbook/)
- S. Staab, R. Struder (Eds.), Handbook on Ontologies, Springer
- A. Gómez-Pérez. Ontological Engineering. Tutorial at IJCAI99 *[http : //www.ontology.org](http://www.ontology.org)*
- Protégé *[http : //protege.stanford.edu/](http://protege.stanford.edu/)*