

# An Introduction to Ontologies and Ontology Extraction

M. Thomazo

# Goal of the Sequence

- ▶ introduce the notion of *ontology*
- ▶ discuss the challenges of automatic ontology construction
- ▶ present an application of ontology (ontology based data access).

# Today's Goal

1. introduce the notion of ontology
2. outline the challenges occurring in automatic ontology construction.

# Ontologies

An **ontology** is a **formal conceptualization** of a domain of interest. Ontologies can be seen as **logical theories**, thereby making knowledge available for machine processing.

# Ontologies

An **ontology** is a **formal conceptualization** of a domain of interest. Ontologies can be seen as **logical theories**, thereby making knowledge available for machine processing.

An ontology defines the **terminology** (vocabulary) of the domain and the **semantics relationships** between terms.

Example (family domain)

- ▶ Terms: parent, mother, sister, sibling, ...
- ▶ Relationships between terms: “mother” is a subclass of “parent”, “sister” is both in the domain and in the range of “has sibling”, “parent” is the disjoint union of “father” and “mother” ...

# Reasons for Using Ontologies

- ▶ **Standardize the terminology** of an application domain : make it easy to share information – well-defined syntax and formal logic-based semantics (i.e. meaning)
  - ▶ complex industrial systems description, scientific knowledge (medicine, life science...)

# Reasons for Using Ontologies

- ▶ **Standardize the terminology** of an application domain : make it easy to share information – well-defined syntax and formal logic-based semantics (i.e. meaning)
  - ▶ complex industrial systems description, scientific knowledge (medicine, life science...)
- ▶ Present an **intuitive and unified view of data sources**: make it easy to formulate queries
  - ▶ data integration, semantic web

# Reasons for Using Ontologies

- ▶ **Standardize the terminology** of an application domain : make it easy to share information – well-defined syntax and formal logic-based semantics (i.e. meaning)
  - ▶ complex industrial systems description, scientific knowledge (medicine, life science...)
- ▶ Present an **intuitive and unified view of data sources**: make it easy to formulate queries
  - ▶ data integration, semantic web
- ▶ Support **automated reasoning**: logical inferences allow us to take advantage of implicit knowledge to answer queries – **computational aspects** can be studied to design **ontology languages** and **tools** that allow for **efficient** reasoning



# Ontology Languages

- ▶ W3C standards: RDFS (RDF Schema) and OWL (Web Ontology Language);
  - ▶ RDFS has low expressivity
  - ▶ OWL Full is undecidable
- ▶ Ontology languages design: **trade-off between expressive power and complexity of reasoning**
- ▶ The formal basis of OWL is first-order logic (FOL).  
The most prominent fragments of FOL in this setting are
  - ▶ **Description Logics** (integrated in OWL 2 profiles)
    - ▶ from the knowledge representation community
  - ▶ **Existential Rules**/tuple-generating dependencies
    - ▶ from the database community

# Description Logics: Syntax

## Basic building blocks

- ▶ atomic **concepts** (unary predicates)
  - ▶ Mother, Sister ...
- ▶ atomic **roles** (binary predicates)
  - ▶ hasChild, isMarriedTo ...
- ▶ **individuals** (constants)
  - ▶ *alice*, *bob* ...

# Description Logics: Syntax

## Basic building blocks

- ▶ atomic **concepts** (unary predicates)
  - ▶ Mother, Sister ...
- ▶ atomic **roles** (binary predicates)
  - ▶ hasChild, isMarriedTo ...
- ▶ **individuals** (constants)
  - ▶ *alice*, *bob* ...

## Complex concepts

- ▶ **concept constructors**:  $\neg C$ ,  $C \sqcap D$ ,  $C \sqcup D$ ,  $\exists R.C$  ...
  - ▶  $\text{Mother} \sqcup \text{Father}$  : “mothers or fathers”
  - ▶  $\text{Mother} \sqcap \neg \exists \text{hasChild.Male}$  : “mothers who don't have any male child”

## Complex roles

- ▶ **role constructors**:  $R^{-}$  (inverse),  $R \circ S$  (composition) ...

# Description Logics: Syntax

DL knowledge base = TBox (ontology) + ABox (data)

TBox (terminological box) specifies knowledge at intensional level

- ▶ describes general knowledge about the domain
- ▶ defines a set of conceptual elements (concepts, roles) and states constraints describing the relationships between them

ABox (assertional box) specifies knowledge at extensional level

- ▶ contains facts about specific individuals
- ▶ specifies a set of instances of the conceptual elements described at the intensional level

Note: the term ontology is sometimes used to refer to the whole knowledge base rather than to the TBox alone.

# Description Logics: Syntax

The TBox contains **concept inclusions**, **role inclusions** and possibly **properties** about roles (transitivity, functionality...).

- ▶  $\text{Mother} \sqsubseteq \text{Parent}$  : “all mothers are parents”
- ▶  $\text{Spouse} \sqsubseteq \exists \text{isMarriedTo}$  : “spouses are married to someone”
- ▶  $\text{hasParent} \sqsubseteq \text{hasChild}^{-}$  : “if  $x$  has parent  $y$ , then  $y$  has child  $x$ ”

# Description Logics: Syntax

The TBox contains **concept inclusions**, **role inclusions** and possibly **properties** about roles (transitivity, functionality...).

- ▶  $\text{Mother} \sqsubseteq \text{Parent}$  : “all mothers are parents”
- ▶  $\text{Spouse} \sqsubseteq \exists \text{isMarriedTo}$  : “spouses are married to someone”
- ▶  $\text{hasParent} \sqsubseteq \text{hasChild}^{-}$  : “if  $x$  has parent  $y$ , then  $y$  has child  $x$ ”

The ABox contains **concept assertions** and **role assertions**.

- ▶  $\text{Mother}(\text{alice})$  : “alice is a mother”
- ▶  $\text{hasParent}(\text{bob}, \text{alice})$  : “bob has parent alice”

# Description Logics: Semantics

- ▶ Declarative, **model-theoretic semantics**:
  - ▶ maps symbolic representations to entities of an abstraction of the real-world (interpretation)
  - ▶ notion of truth that allows us to determine whether a symbolic expression is true in the world under consideration (model)
- ▶ Not procedural semantics: not defined by how certain algorithms behave
- ▶ Results depend only on the semantics, not on the syntactic representation: semantically equivalent knowledge bases lead to the same results

# Description Logics: Semantics

Interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

- ▶  $\Delta^{\mathcal{I}}$  is a non-empty set called **domain**
- ▶  $\cdot^{\mathcal{I}}$  is a function which associates
  - ▶ each constant  $a$  with an element  $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - ▶ each atomic concept  $A$  with a unary relation  $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - ▶ each atomic role  $R$  with a binary relation  $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$



# Description Logics: Semantics

Interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

- ▶  $\Delta^{\mathcal{I}}$  is a non-empty set called **domain**
- ▶  $\cdot^{\mathcal{I}}$  is a function which associates
  - ▶ each constant  $a$  with an element  $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - ▶ each atomic concept  $A$  with a unary relation  $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - ▶ each atomic role  $R$  with a binary relation  $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Example:

$\Delta^{\mathcal{I}} = \{a, b, c, d, e, f, g\}$

$alice^{\mathcal{I}} = a, bob^{\mathcal{I}} = b$

$Mother^{\mathcal{I}} = \{a, c\}$

$Father^{\mathcal{I}} = \{b, e\}$

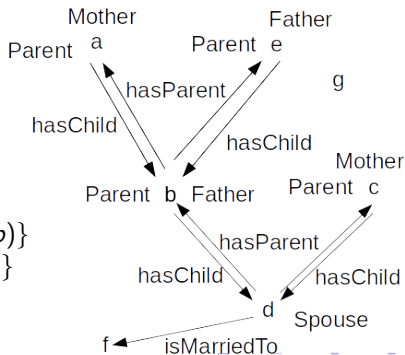
$Parent^{\mathcal{I}} = \{a, b, c, e\}$

$Spouse^{\mathcal{I}} = \{d\}$

$hasParent^{\mathcal{I}} = \{(b, a), (b, e), (d, c), (d, b)\}$

$hasChild^{\mathcal{I}} = \{(a, b), (e, b), (c, d), (b, d)\}$

$isMarriedTo^{\mathcal{I}} = \{(d, f)\}$



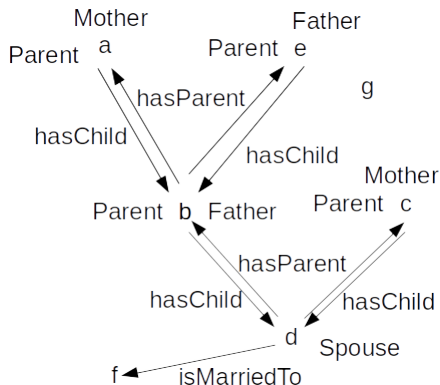
# Description Logics: Semantics

The function  $\cdot^{\mathcal{I}}$  is extended to complex concepts and roles to formalize the meaning of the constructors:

- ▶  $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$  and  $\perp^{\mathcal{I}} = \emptyset$
- ▶  $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- ▶  $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- ▶  $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- ▶  $(R^{-})^{\mathcal{I}} = \{(u, v) \mid (v, u) \in R^{\mathcal{I}}\}$
- ▶  $(\exists R.C)^{\mathcal{I}} = \{u \mid \text{there exists } (u, v) \in R^{\mathcal{I}} \text{ such that } v \in C^{\mathcal{I}}\}$
- ▶  $(\forall R.C)^{\mathcal{I}} = \{u \mid \text{for every } v, \text{ if } (u, v) \in R^{\mathcal{I}} \text{ then } v \in C^{\mathcal{I}}\}$
- ▶ ...

# Description Logics: Semantics

## Example



$$(\neg \text{Parent})^{\mathcal{I}} = ?$$

$$(\exists \text{hasParent}.\top)^{\mathcal{I}} = ?$$

$$(\text{isMarriedTo}^-)^{\mathcal{I}} = ?$$

$$(\text{Spouse} \sqcup \text{Mother})^{\mathcal{I}} = ?$$

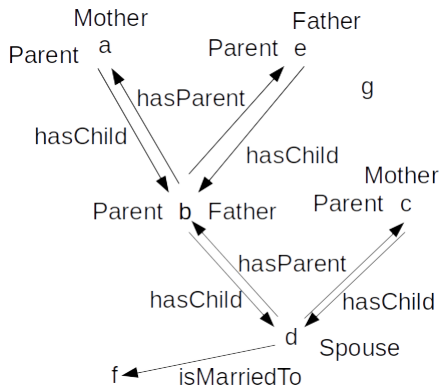
$$(\forall \text{hasChild}.\text{Spouse})^{\mathcal{I}} = ?$$

$$((\forall \text{hasChild}.\text{Spouse}) \sqcap (\exists \text{hasChild}.\top))^{\mathcal{I}} = ?$$

$$(\text{Mother} \sqcap (\exists \text{hasChild}.\exists \text{hasChild}^-. \exists \text{hasParent}.\text{Father}))^{\mathcal{I}} = ?$$

# Description Logics: Semantics

## Example



$$(\neg \text{Parent})^{\mathcal{I}} = \{d, f, g\}$$

$$(\exists \text{hasParent}.\top)^{\mathcal{I}} = \{b, d\}$$

$$(\text{isMarriedTo}^-)^{\mathcal{I}} = \{(f, d)\}$$

$$(\text{Spouse} \sqcup \text{Mother})^{\mathcal{I}} = \{a, c, d\}$$

$$(\forall \text{hasChild}.\text{Spouse})^{\mathcal{I}} = \{b, c, d, f, g\}$$

$$((\forall \text{hasChild}.\text{Spouse}) \sqcap (\exists \text{hasChild}.\top))^{\mathcal{I}} = \{b, c\}$$

$$(\text{Mother} \sqcap (\exists \text{hasChild}.\exists \text{hasChild}^-. \exists \text{hasParent}.\text{Father}))^{\mathcal{I}} = \{c\}$$

# Description Logics: Semantics

## Satisfaction of TBox axioms

- ▶  $\mathcal{I}$  satisfies a concept inclusion  $C \sqsubseteq D$ , written  $\mathcal{I} \models C \sqsubseteq D$ , if  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- ▶  $\mathcal{I}$  satisfies a role inclusion  $R \sqsubseteq S$ , written  $\mathcal{I} \models R \sqsubseteq S$ , if  $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- ▶  $\mathcal{I}$  satisfies  $(\text{func } R)$ , written  $\mathcal{I} \models (\text{func } R)$ , if  $R^{\mathcal{I}}$  is a functional relation
- ▶ ...

# Description Logics: Semantics

## Satisfaction of TBox axioms

- ▶  $\mathcal{I}$  satisfies a concept inclusion  $C \sqsubseteq D$ , written  $\mathcal{I} \models C \sqsubseteq D$ , if  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- ▶  $\mathcal{I}$  satisfies a role inclusion  $R \sqsubseteq S$ , written  $\mathcal{I} \models R \sqsubseteq S$ , if  $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- ▶  $\mathcal{I}$  satisfies  $(\text{func } R)$ , written  $\mathcal{I} \models (\text{func } R)$ , if  $R^{\mathcal{I}}$  is a functional relation
- ▶ ...

## Satisfaction of ABox assertions

- ▶  $\mathcal{I}$  satisfies a concept assertion  $C(a)$ , written  $\mathcal{I} \models C(a)$ , if  $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- ▶  $\mathcal{I}$  satisfies a role assertion  $R(a, b)$ , written  $\mathcal{I} \models R(a, b)$ , if  $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

# Description Logics: Semantics

## Satisfaction of TBox axioms

- ▶  $\mathcal{I}$  satisfies a concept inclusion  $C \sqsubseteq D$ , written  $\mathcal{I} \models C \sqsubseteq D$ , if  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- ▶  $\mathcal{I}$  satisfies a role inclusion  $R \sqsubseteq S$ , written  $\mathcal{I} \models R \sqsubseteq S$ , if  $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- ▶  $\mathcal{I}$  satisfies (*func*  $R$ ), written  $\mathcal{I} \models (\text{func } R)$ , if  $R^{\mathcal{I}}$  is a functional relation
- ▶ ...

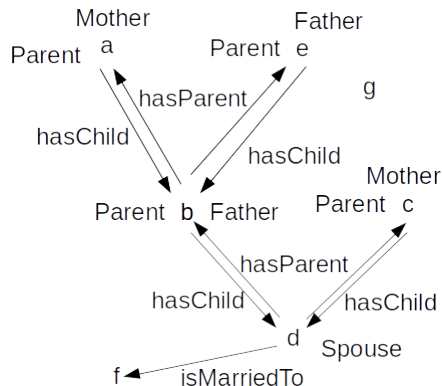
## Satisfaction of ABox assertions

- ▶  $\mathcal{I}$  satisfies a concept assertion  $C(a)$ , written  $\mathcal{I} \models C(a)$ , if  $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- ▶  $\mathcal{I}$  satisfies a role assertion  $R(a, b)$ , written  $\mathcal{I} \models R(a, b)$ , if  $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

**Open-world assumption:** the absence of an assertion does not mean that it is false (different from closed-world assumption used for databases)

# Description Logics: Semantics

## Example



$\mathcal{I} \models \text{Mother} \sqsubseteq \text{Parent} ?$

$\mathcal{I} \models \exists \text{hasChild}.\top \sqsubseteq \exists \text{hasParent}.\top ?$

$\mathcal{I} \models \text{Mother} \sqsubseteq \neg \text{Father} ?$

$\mathcal{I} \models (\text{func } \text{hasChild}) ?$

$\mathcal{I} \models \text{hasParent}(\text{bob}, \text{alice}) ?$

$\mathcal{I} \models \exists \text{hasChild} . (\text{Father} \sqcap \exists \text{hasChild} . \text{Spouse})(\text{alice}) ?$

$\mathcal{I} \models \forall \text{hasChild} . (\text{Father} \sqcap \forall \text{isMarriedTo} . \text{Spouse})(\text{alice}) ?$



# Description Logics: Semantics

## Models

- ▶  $\mathcal{I}$  is a model of a TBox  $\mathcal{T}$  if it satisfies every axiom in  $\mathcal{T}$
- ▶  $\mathcal{I}$  is a model of an ABox  $\mathcal{A}$  if it satisfies every assertion in  $\mathcal{A}$
- ▶  $\mathcal{I}$  is a model of a KB  $\langle \mathcal{T}, \mathcal{A} \rangle$  if it is a model of  $\mathcal{T}$  and  $\mathcal{A}$
- ▶ Two KBs are **equivalent** if they have the same models

# Description Logics: Semantics

## Models

- ▶  $\mathcal{I}$  is a model of a TBox  $\mathcal{T}$  if it satisfies every axiom in  $\mathcal{T}$
- ▶  $\mathcal{I}$  is a model of an ABox  $\mathcal{A}$  if it satisfies every assertion in  $\mathcal{A}$
- ▶  $\mathcal{I}$  is a model of a KB  $\langle \mathcal{T}, \mathcal{A} \rangle$  if it is a model of  $\mathcal{T}$  and  $\mathcal{A}$
- ▶ Two KBs are **equivalent** if they have the same models

## Satisfiability

- ▶ A KB  $\langle \mathcal{T}, \mathcal{A} \rangle$  is **satisfiable**, or **consistent**, if it has at least one model
- ▶ A concept  $C$  is satisfiable w.r.t. a TBox  $\mathcal{T}$  if there exists a model  $\mathcal{I}$  of  $\mathcal{T}$  such that  $C^{\mathcal{I}} \neq \emptyset$

# Description Logics: Semantics

## Models

- ▶  $\mathcal{I}$  is a model of a TBox  $\mathcal{T}$  if it satisfies every axiom in  $\mathcal{T}$
- ▶  $\mathcal{I}$  is a model of an ABox  $\mathcal{A}$  if it satisfies every assertion in  $\mathcal{A}$
- ▶  $\mathcal{I}$  is a model of a KB  $\langle \mathcal{T}, \mathcal{A} \rangle$  if it is a model of  $\mathcal{T}$  and  $\mathcal{A}$
- ▶ Two KBs are **equivalent** if they have the same models

## Satisfiability

- ▶ A KB  $\langle \mathcal{T}, \mathcal{A} \rangle$  is **satisfiable**, or **consistent**, if it has at least one model
- ▶ A concept  $C$  is satisfiable w.r.t. a TBox  $\mathcal{T}$  if there exists a model  $\mathcal{I}$  of  $\mathcal{T}$  such that  $C^{\mathcal{I}} \neq \emptyset$

## Entailment

- ▶ A TBox  $\mathcal{T}$  entails an axiom  $\alpha$ , written  $\mathcal{T} \models \alpha$ , if every model of  $\mathcal{T}$  satisfies  $\alpha$
- ▶ A KB  $\langle \mathcal{T}, \mathcal{A} \rangle$  entails an assertion  $\alpha$ , written  $\langle \mathcal{T}, \mathcal{A} \rangle \models \alpha$ , if every model of  $\langle \mathcal{T}, \mathcal{A} \rangle$  satisfies  $\alpha$

# Relationship between DLs and OWL

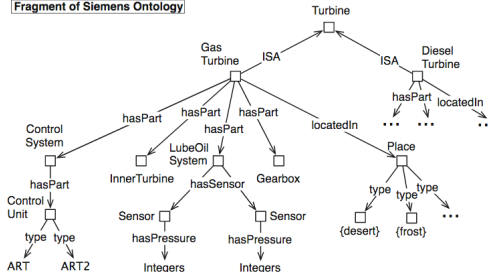
Mapping (sub-languages of) OWL to equivalent DLs provides a well-defined semantics and allows us to use results of DL research

- ▶ Semantics of OWL 2 is directly based on DLs
- ▶ Complexity results, algorithms and implemented reasoners
- ▶ OWL 2 profiles (OWL 2 EL, OWL 2 QL, and OWL 2 RL) correspond to DL languages with interesting computational properties, targeted towards a specific use

# Examples of Applications of Ontologies

## Ontologies for Industry

Fragment of Siemens Ontology



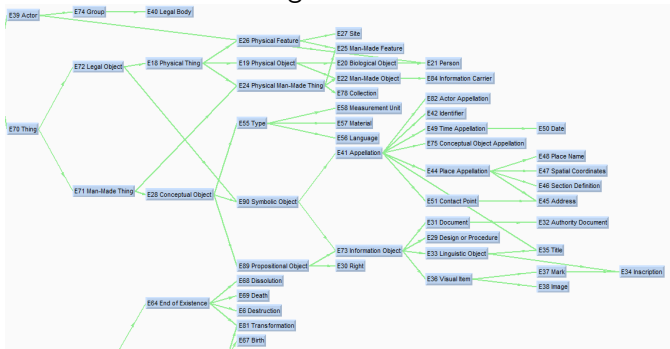
From: How Semantic Technologies Can Enhance Data Access at Siemens Energy, Kharlamov et al., ISWC 2014

- ▶ Energy sector: Optique EU project (several universities involved)
  - ▶ Siemens: turbines diagnostics
  - ▶ StatOil: find exploitable accumulations of oil or gas
- ▶ Aeronautics sector
  - ▶ Collaboration between Thales and Univ. Paris Sud on ontology-based solutions for avionics maintenance
  - ▶ NASA Air Traffic Management Ontology

# Examples of Applications of Ontologies

## Ontologies for Public Policies

- ▶ Collaboration between Sapienza Univ. & Italian Department of Treasury on ontology-based data management of public debt
- ▶ CIDOC CRM (Comité International pour la DOcumentation Conceptual Reference Model): ontology for concepts and information in cultural heritage and museum documentation



Part of CIDOC CRM Class Hierarchy from [www.cidoc-crm.org](http://www.cidoc-crm.org)

# Examples of Applications of Ontologies

## Medical Ontologies

Body structure	Preferred Name	Traumatic epistaxis
Clinical finding	Synonyms	Traumatic epistaxis (disorder)
Administrative statuses	ID	<a href="http://purl.bioontology.org/ontology/SNOMEDCT/232356000">http://purl.bioontology.org/ontology/SNOMEDCT/232356000</a>
Adverse incident outcome categories	Active	1
Bleeding	altLabel	Traumatic epistaxis (disorder)
Accidental hemorrhage during medical ca	CASE SIGNIFICANCE ID	9000000000000448009
Acute benign hemorrhagic glomerulonephr	CTV3ID	X00IF
Acute hemorrhagic cholecystitis	cul	C0339817
Adrenal hemorrhage	DEFINITION STATUS ID	900000000000073002
Anastomotic bleeding	Due to	<a href="#">Traumatic injury</a>
Ascorbic acid deficiency with hemorrhage	Effective time	20180131
Bleeding cervix	Has associated morphology	<a href="#">Hemorrhage</a>
Bleeding from anus	Has finding site	<a href="#">Structure of blood vessel of internal nose</a>
Bleeding from breast		
Bleeding from ear		
Bleeding from larynx		
Bleeding from mouth		
Bleeding from nose		
Neonatal epistaxis		
On examination - epistaxis		
Perinatal epistaxis		
Post-surgical epistaxis		
<b>Traumatic epistaxis</b>		
Bleeding from urethra		

- ▶ SNOMED CT: general medical ontology ( > 350 000 concepts)
  - ▶ multilingual, mapped to other international standards
  - ▶ used for recording medical information : information sharing, decision-making assistance systems, gathering data for clinical research, monitoring population health and clinical practices...
- ▶ NCI (National Cancer Institute Thesaurus), FMA (Foundational Model of Anatomy), GO (Gene Ontology) ...

# Examples of Applications of Ontologies

## Ontologies for Life Sciences

- ▶ Bioportal repository contains hundreds of ontologies about biology and chemistry (<http://bioportal.bioontology.org/>)

The screenshot shows the BioPortal website interface. At the top is a dark blue header with the BioPortal logo and navigation links: Ontologies, Search, Annotator, Recommender, Mappings, and Resource Index. On the right of the header are 'Login' and 'Support' buttons. Below the header, the main content area is titled 'Browse' with a subtitle 'Browse the library of ontologies'. A search bar is located at the top of the main area, showing 'Showing 19 of 995' and 'Sort: Popular'. On the left side, there are three filter panels: 'Submit New Ontology' (a button), 'Entry Type' (with 'Ontology (19)' selected and 'Ontology View (0)' unselected), and 'Uploaded in the Last' (a dropdown menu). Below these is a 'Category' panel with a list of categories: 'All Organisms (29)', 'Anatomy (68)', 'Animal Development (12)', 'Animal Gross Anatomy' (selected), 'Arabidopsis (3)', and 'Biological Process (50)'. The main list of ontologies includes: 1. 'Drosophila Gross Anatomy Ontology (FB-BT)' with 3 products and 11,564 classes, uploaded 11/26/19. 2. 'Zebrafish Anatomy and Development Ontology (ZFA)' with 2 products, 4 mappings, and 3,213 classes, uploaded 12/14/19. 3. 'Spider Anatomy Ontology (SPD)' with 2 products and 832 classes, uploaded 9/24/19. 4. 'Tick Gross Anatomy Ontology (TADS)' with 2 products and 628 classes, uploaded 8/21/15. 5. 'Teleost Anatomy Ontology (TAO)' with 2 products, 1 mapping, and 3,428 classes. At the bottom right, there are navigation icons for back, forward, and search.



# Examples of Applications of Ontologies

## Knowledge Graphs

Knowledge graphs : flexible tool to represent knowledge



## Albert Einstein



Theoretical physicist

Albert Einstein was a German-born theoretical physicist who developed the theory of relativity, one of the two pillars of modern physics. His work is also known for its influence on the philosophy of science.

[Wikipedia](#)

**Born:** March 14, 1879, [Ulm, Germany](#)

**Died:** April 18, 1955, [Princeton Medical Center, New Jersey, United States](#)

**Education:** [University of Zurich \(1905\)](#), [ETH Zürich \(1896–1900\)](#), [MORE](#)

**Spouse:** [Elsa Einstein](#) (m. 1919–1936), [Mileva Marić](#) (m. 1903–1919)

# Ontology Editors and Reasoners

A lot of reasoners and tools and libraries for developing ontologies have been implemented. Reasoners support various ontology languages and reasoning tasks, and implement various algorithms.

- ▶ List of DL reasoners:

<http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/>

- ▶ List of OWL implementations (reasoners, editors, API...):

<http://www.w3.org/2001/sw/wiki/OWL/Implementations>

# Hands-on Session with Protégé

Protégé ontology editor: <http://protege.stanford.edu/>

- ▶ Free
- ▶ Open-source
- ▶ Lots of plugins
- ▶ Integrate several DL reasoners
  - ▶ check ontology consistency
  - ▶ infer new subclasses relationships
  - ▶ query the ontology
  - ▶ explain some inferences

# Hands-on Session with Protégé

The screenshot shows the Protégé ontology editor interface. The browser address bar at the top displays the URL `pizza (http://www.co-ode.org/ontologies/pizza/2.0.0)`. A callout bubble points to the 'View selection' button in the address bar. The main interface is divided into several panels:

- Ontology header:** Displays the **Ontology IRI** (`http://www.co-ode.org/ontologies/pizza`) and the **Ontology Version IRI** (`http://www.co-ode.org/ontologies/pizza/2.0.0`).
- Annotations:** A list of annotations for the ontology. A callout bubble points to the 'Define ontology prefix' button. The annotations include:
  - `rdfs:label` [type: xsd:string] pizza
  - `dc:title` [language: en] pizza
  - `dc:description` [language: en] An ontology about pizzas and their toppings.  
This is an example ontology that contains all constructs required for the various versions of the Pizza Tutorial run by Manchester University (see <http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protégé-owl-tutorial>).
  - `dc:license` [type: xsd:string] Creative Commons Attribution 3.0 (CC BY 3.0)
  - `dc:contributor` Alan Rector
  - `dc:contributor` Chris Wroe
  - `dc:contributor`
- Ontology metrics:** A table showing various metrics for the ontology.
- Class axioms:** A table showing class axioms.
- Object property axioms:** A table showing object property axioms.
- Imported ontologies:** A section for listing imported ontologies, with sub-sections for Direct Imports and Indirect Imports.

Metric	Count
Axiom	801
Logical axiom count	322
Declaration axioms count	120
Class count	100
Object property count	8
Data property count	0
Individual count	5
Annotation Property count	12

Axiom	Count
SubClassOf	259
EquivalentClasses	15
DisjointClasses	14
GCI count	0
Hidden GCI Count	2

Axiom	Count
SubObjectPropertyOf	4
EquivalentObjectProperties	0
InverseObjectProperties	3
DisjointObjectProperties	0
FunctionalObjectProperty	4

# Hands-on Session with Protégé

The screenshot shows the Protégé ontology editor interface. The left pane displays the class hierarchy, with 'Cajun' selected under 'NamedPizza'. A callout bubble points to 'Cajun' with the text 'Select class to get description'. The right pane shows the 'Annotations: Cajun' tab, listing various annotations. Below it, the 'Description: Cajun' tab is active, showing the class description. A callout bubble points to the description with the text 'Expressions of super classes using Manchester syntax'. Another callout bubble points to the 'Cajun' class name in the description with the text 'Explain'.

Active ontology: x Entities x Individuals by class x DL Query x

Annotation properties: Classes Object properties Datatypes Individuals Data properties

Class hierarchy: Cajun

Annotations: Cajun

Annotations:

- `rdfs:label` [language: en] Cajun
- `rdfs:label` [language: pt] Cajun
- `skos:prefLabel` [language: en] Cajun
- `skos:altLabel` [language: en] Cajun Pizza

Description: Cajun

Equivalent To

SubClass Of

- `hasTopping` only (MozzarellaTopping or OnionTopping or PeperonataTopping or PrawnsTopping or TabascoPepperSauceTopping or TomatoTopping)
- `hasTopping` some MozzarellaTopping
- `hasTopping` some OnionTopping
- `hasTopping` some PeperonataTopping
- `hasTopping` some PrawnsTopping
- `hasTopping` some TabascoPepperSauceTopping
- `hasTopping` some TomatoTopping
- `NamedPizza`

General

`Cajun`  $\sqsubseteq$   $\exists$ hasTopping.TomatoTopping

SubClass Of (Anonymous Ancestor)

- `hasBase` some PizzaBase

Instances

Target for Key

# Hands-on Session with Protégé

The screenshot displays the Protégé ontology editor interface for the 'pizza' ontology (http://www.co-ode.org/ontologies/pizza/2.0.0). The interface is divided into several panes:

- Top Bar:** Shows the ontology name 'pizza' and a search bar.
- Left Pane:** Contains the 'Object property hierarchy: hasIngredient' tree. The hierarchy is as follows:
  - owl:topObjectProperty
    - hasCountryOfOrigin
    - hasIngredient
      - hasBase
      - hasTopping
    - hasSpiciness
    - isIngredientOf
      - isBaseOf
      - isToppingOf
- Annotations: hasIngredient:** Shows an annotation: `rdfs:comment` [language: en] with the text 'NB Transitive - the ingredients of ingredients are ingredients of the whole'.
- Characteristics: hasIngredient:** A list of checkboxes for property characteristics. The 'Transitive' checkbox is checked.
  - ☐ Functional
  - ☐ Inverse functional
  - ☒ Transitive
  - ☐ Symmetric
  - ☐ Asymmetric
  - ☐ Reflexive
  - ☐ Irreflexive
- Description: hasIngredient:** A list of property relationships with hand-drawn callouts:
  - Equivalent To:** Empty.
  - SubProperty Of:** Empty.
  - Inverse Of:** `isIngredientOf` (callout: `hasIngredient ⊑ isIngredientOf`).
  - Domains (intersection):** `Food` (callout: `∃ hasIngredient ⊑ Food`).
  - Ranges (intersection):** `Food`.
  - Disjoint With:** Empty.
  - SuperProperty Of (Chain):** Empty.

# Hands-on Session with Protégé

The screenshot displays the Protégé ontology editor interface. At the top, the browser address bar shows the URL `http://www.co-ode.org/ontologies/pizza/pizza.owl#America`. The main window is divided into several panes:

- Left Pane:** Contains a tree view of the ontology. The 'America' class is selected, and its subclasses are listed: England, France, Germany, and Italy.
- Top Pane:** Shows the 'Active ontology' and 'Entities' tabs. The 'Individuals by class' tab is active, displaying the 'America' class.
- Right Pane:** Contains two sub-panes: 'Annotations: America' and 'Property assertions: America'. The 'Annotations: America' pane is currently empty. The 'Property assertions: America' pane shows a list of property assertions, including 'Country' and 'owl:Thing'.

The bottom of the interface features a navigation bar with various icons for navigating through the ontology.

# Hands-on Session with Protégé

The screenshot displays the Protégé ontology editor interface. At the top, the browser address bar shows the URL `http://www.co-ode.org/ontologies/pizza/2.0.0/`. The main interface is divided into several panes:

- Class hierarchy:** Shows a tree view with `owl:Thing` as the root class.
- DL query:** Contains a text area with the query `SpicyPizza and NonVegetarianPizza`. Below the text area are buttons for `Execute` and `Add to ontology`.
- Query results:** Displays a list of subclasses (7 of 7) for the query. The results are:
  - AmericanHot
  - Cajun
  - CheesyVegetableTopping
  - IceCream
  - PolloAdAstra
  - SloppyGiuseppe
  - owl:Nothing** (highlighted)
- Query for:** A panel on the right with checkboxes for:
  - ☐ Direct superclasses
  - ☐ Superclasses
  - ☐ Equivalent classes
  - ☐ Direct subclasses
  - ☒ Subclasses
  - ☐ Instances
- Result filters:** A panel on the right with checkboxes for:
  - ☒ Display owl:Thing (in superclass results)
  - ☒ Display owl:Nothing (in subclass results)



# Hands-on Session with Protégé

## Getting Started with the Pizza Ontology

- ▶ Download the Pizza ontology:  
`http://protege.stanford.edu/ontologies/pizza/pizza.owl`
- ▶ Open it with Protégé (File > Open).
- ▶ Identify concepts (classes) and roles (object properties).
- ▶ Identify relationships between them, translate them into DL syntax (complex concepts, subconcepts, disjoint concepts...).
- ▶ Select and start reasoner
- ▶ Check some inferences explanations

# Hands-on Session with Protégé

## Creating an Ontology

- ▶ Create an ontology (File > New) with IRI “http://small-onto” and save it (File > Save as...) in RDF/XML syntax.
- ▶ Express the following statements as DL axioms, then add them to your ontology in Protégé.
  - ▶ Mammals are animals that produce milk
  - ▶ Cats, cows, pigs and platypus are mammals
  - ▶ Birds are animals that do not produce milk
  - ▶ Birds and platypus lay eggs
  - ▶ Cows eat only plants
  - ▶ Cats and platypus are carnivorous
  - ▶ Pigs eat both plants and meat
  - ▶ Animals that only eat plants are herbivorous
  - ▶ Carnivorous are animals that eat meat
  - ▶ Animals that eat both plants and meat are omnivorous
  - ▶ Meat and plants are disjoint
  - ▶ Something that is eaten is food

# Ontology Construction

Usually a manual endeavor:

- ▶ requires knowledge from the domain
- ▶ requires knowledge about formal ontologies.
- ▶ ontologies may be used:
- ▶ hard to debug.

Tools to help manual ontology design:

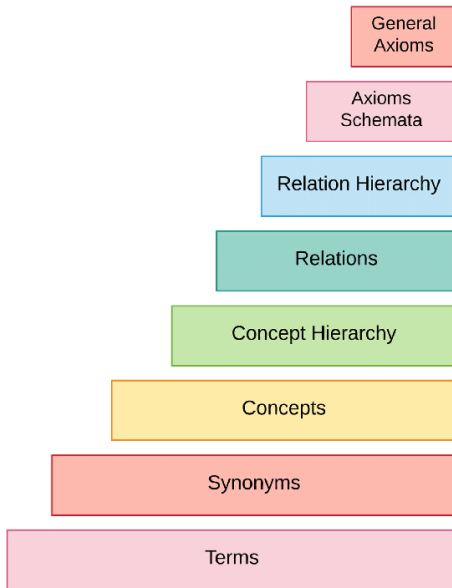
- ▶ satisfiability checks;
- ▶ entailment computation;
- ▶ module extraction;
- ▶ ontology alignment.

# Ontology Construction: Towards Automatic Construction

Methods to (semi-)automatically build ontologies:

- ▶ different possible types of inputs:
  - ▶ (semi-)structured data (databases, knowledge graphs)
  - ▶ unstructured data (text)
- ▶ different goals of learning: "ontology learning layer cake"

# Ontology Learning Layer Cake



# The different layers

- ▶ terms: river, country, city, town, ...

# The different layers

- ▶ terms: river, country, city, town, ...
- ▶ synonyms: city, town

# The different layers

- ▶ terms: river, country, city, town, ...
- ▶ synonyms: city, town
- ▶ concepts: City



# The different layers

- ▶ terms: river, country, city, town, ...
- ▶ synonyms: city, town
- ▶ concepts: City
- ▶ concept hierarchy: Capital  $\sqsubseteq$  City

# The different layers

- ▶ terms: river, country, city, town, ...
- ▶ synonyms: city, town
- ▶ concepts: City
- ▶ concept hierarchy: Capital  $\sqsubseteq$  City
- ▶ relations: isCapitalOf(domain: City, range: Nation)

# The different layers

- ▶ terms: river, country, city, town, ...
- ▶ synonyms: city, town
- ▶ concepts: City
- ▶ concept hierarchy: Capital  $\sqsubseteq$  City
- ▶ relations: isCapitalOf(domain: City, range: Nation)
- ▶ relation hierarchy: isCapitalOf  $\sqsubseteq$  isLocatedIn

# The different layers

- ▶ terms: river, country, city, town, ...
- ▶ synonyms: city, town
- ▶ concepts: City
- ▶ concept hierarchy: Capital  $\sqsubseteq$  City
- ▶ relations: isCapitalOf(domain: City, range: Nation)
- ▶ relation hierarchy: isCapitalOf  $\sqsubseteq$  isLocatedIn
- ▶ axioms schemata: City  $\sqcap$  River  $\sqsubseteq \perp$

# The different layers

- ▶ terms: river, country, city, town, ...
- ▶ synonyms: city, town
- ▶ concepts: City
- ▶ concept hierarchy: Capital  $\sqsubseteq$  City
- ▶ relations: isCapitalOf(domain: City, range: Nation)
- ▶ relation hierarchy: isCapitalOf  $\sqsubseteq$  isLocatedIn
- ▶ axioms schemata: City  $\sqcap$  River  $\sqsubseteq \perp$
- ▶ general axioms: the rest :)

Could add aligning with existing ontologies.

# Evaluation

Four "types" of evaluation:

- ▶ "Gold standard": comparison with an ideal reference ontology
- ▶ application-based: improvement in the performance of an application using the ontology vs not using it
- ▶ data-driven: suitability between data and the built ontology
- ▶ human.

# Kind of techniques Used

- ▶ pattern-based extraction
  - ▶ reasonable precision
  - ▶ very low recall
- ▶ POS tagging, sentence parsing
  - ▶ ambiguity
- ▶ co-occurrence analysis
  - ▶ good result for concept formation
  - ▶ not appropriate for relation discovery
- ▶ heuristic and conceptual clustering
  - ▶ not appropriate for non-taxonomic axioms
- ▶ inductive logic programming

# A Word on Inductive Logic Programming

Facts:

- ▶  $\text{grandfather}(x, y) \leftarrow \text{father}(x, z), \text{parent}(z, y)$
- ▶  $\text{father}(\text{henry}, \text{jane})$
- ▶  $\text{mother}(\text{jane}, \text{john})$
- ▶  $\text{mother}(\text{jane}, \text{alice})$

Positive examples:

- ▶  $\text{grandfather}(\text{henry}, \text{john})$
- ▶  $\text{grandfather}(\text{henry}, \text{alice})$

Negative Examples:

- ▶  $\text{grandfather}(\text{john}, \text{henry})$
- ▶  $\text{grandfather}(\text{alice}, \text{john})$



# A Word on Inductive Logic Programming

Given  $B$ , facing  $E^+$  and  $E^-$ , one might guess the following rule  $H$ :

- ▶  $\text{parent}(x, y) \leftarrow \text{mother}(x, y)$

Note that  $H$ :

- ▶ is not a logical consequence of what is known
- ▶ allows us to explain  $E^+$
- ▶ is consistent with  $E^-$

Challenge of Inductive Logic Programming: generating "interesting" hypothesis.

# Wrap Up

We have seen:

- ▶ what an ontology is;
- ▶ a very short introduction to description logics as a way to formalize them;
- ▶ some challenges occurring in building them;
- ▶ some techniques used.

# Next Week

- ▶ Close up on AMIE: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases (Galárraga et al.)
- ▶ An example of use of ontologies: ontology-based data access.