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 notin of ontology
2. outline the challenges occuring 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: $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$ ...
  - Mother $\sqcup$ Father : "mothers or fathers"
  - Mother $\sqcap \neg\exists$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...).

- $\mathsf{Mother} \sqsubseteq \mathsf{Parent}$: "all mothers are parents"
- $\mathsf{Spouse} \sqsubseteq \exists \mathsf{isMarriedTo}$: "spouses are married to someone"
- $\mathsf{hasParent} \sqsubseteq \mathsf{hasChild}^-$: "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...).

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

The ABox contains concept assertions and role assertions.

- Mother(*alice*) : "alice is a mother"
- hasParent(*bob*, *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:

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

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

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

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

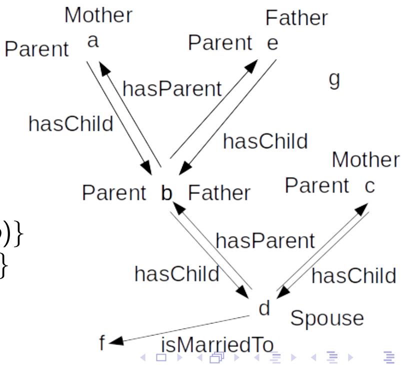
$\text{Parent}^{\mathcal{I}} = \{a, b, c, e\}$

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

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

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

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



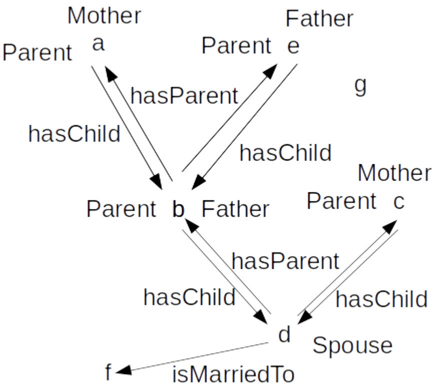
Description Logics: Semantics

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

Description Logics: Semantics

Example



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

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

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

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

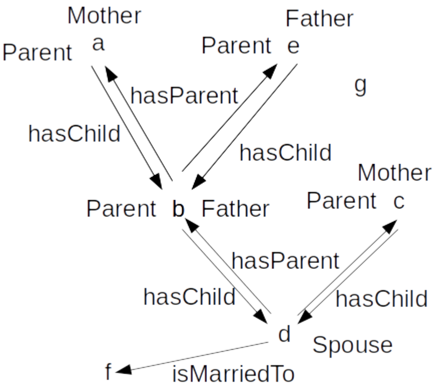
$(\forall \text{hasChild}.\text{Spouse})^{\mathcal{I}} = ?$

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

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

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 $(\mathit{func}\ R)$, written $\mathcal{I} \models (\mathit{func}\ R)$, 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 $(\mathit{func}\ R)$, written $\mathcal{I} \models (\mathit{func}\ R)$, 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 $\mathcal{I} \models (func\ R)$, if $R^{\mathcal{I}}$ is a functional relation
- ...

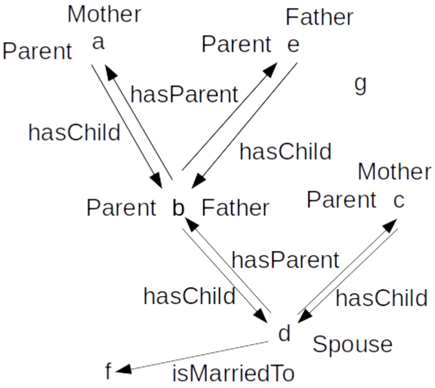
Satisfaction of ABox assertions

- $\mathcal{I}$ 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



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

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

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

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

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

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

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

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

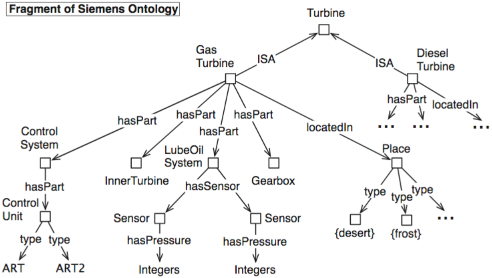
Relationship between DLs and OWL

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



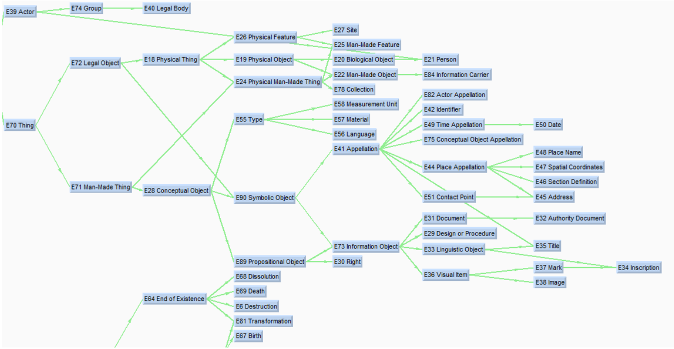
From: How Semantic Technologies Can Enhance Data Access at Siemens Energy, Kharlamov et al., ISWC 2014

- 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

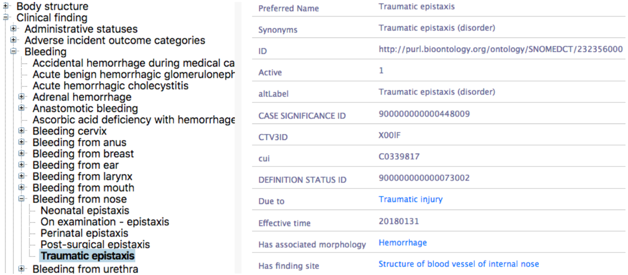
- Collaboration between Sapienza Univ. & Italian Department of Treasury on ontology-based data management of public debt
- CIDOC CRM (Comité International pour la DOCumentation Conceptual Reference Model): ontology for concepts and information in cultural heritage and museum documentation



Part of CIDOC CRM Class Hierarchy from www.cidoc-crm.org

Examples of Applications of Ontologies

Medical Ontologies

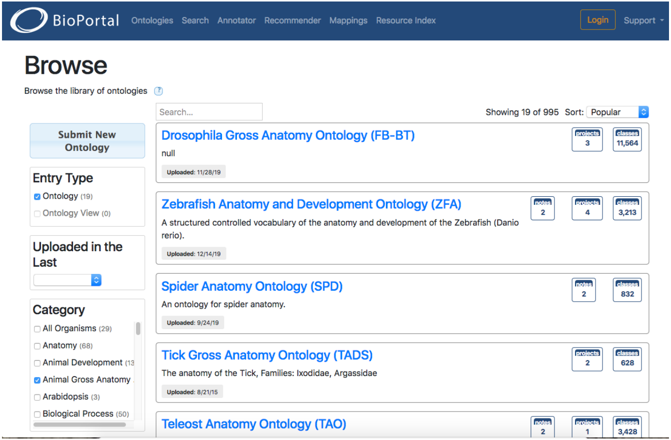


- SNOMED CT: general medical ontology ($>$ 350 000 concepts)
  - multilingual, mapped to other international standards
  - used for recording medical information : information sharing, decision-making assistance systems, gathering data for clinical research, monitoring population health and clinical 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/`)



Examples of Applications of Ontologies

Knowledge Graphs

Knowledge graphs : flexible tool to represent knowledge

WIKIDATA

yago select knowledge

DBpedia

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.

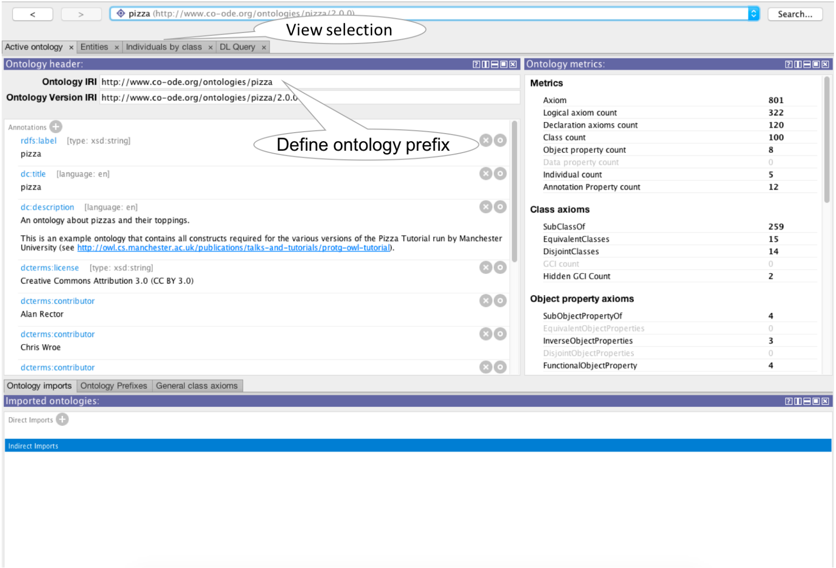
- List of DL reasoners:
  `http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/`
- List of OWL implementations (reasoners, editors, API...):
  `http://www.w3.org/2001/sw/wiki/OWL/Implementations`

Hands-on Session with Protégé

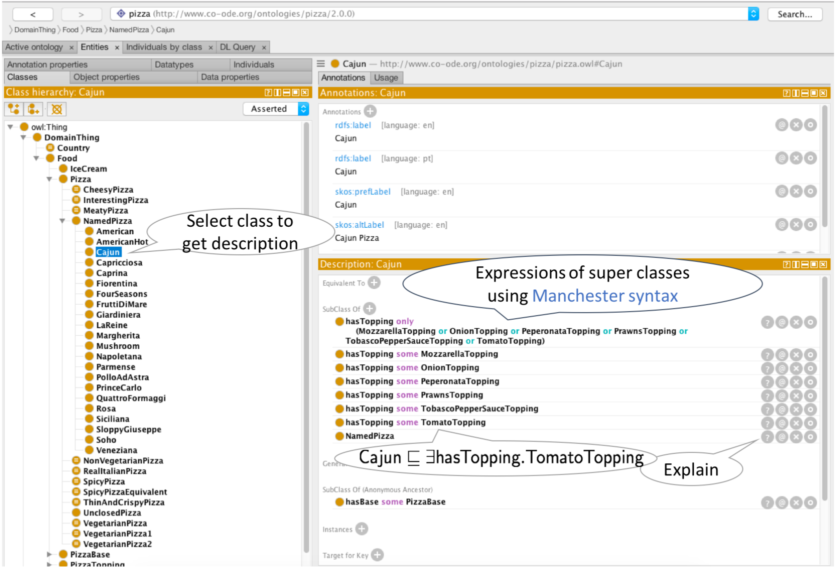
Protégé ontology editor: `http://protege.stanford.edu/`

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

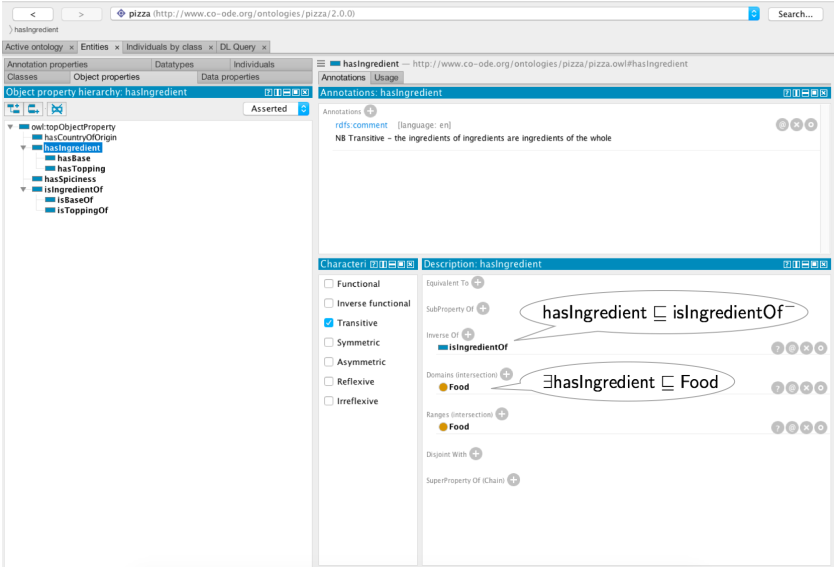
Hands-on Session with Protégé



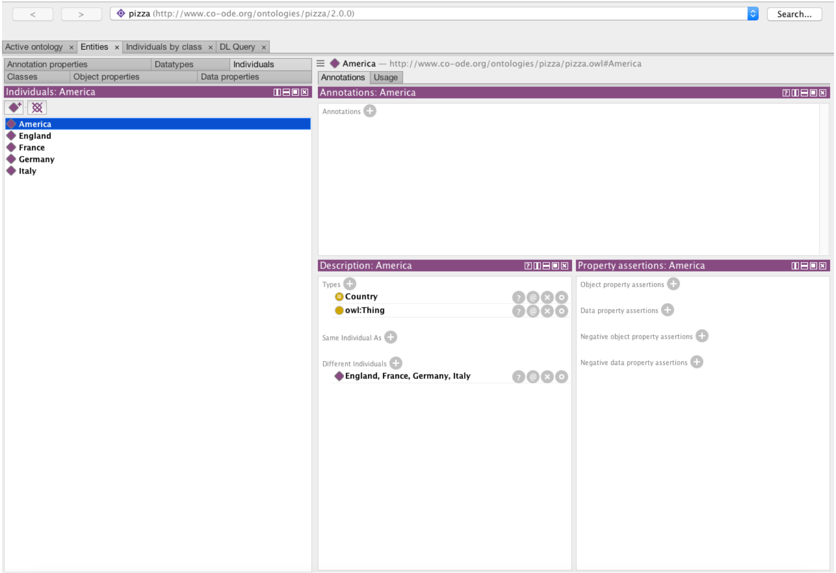
Hands-on Session with Protégé



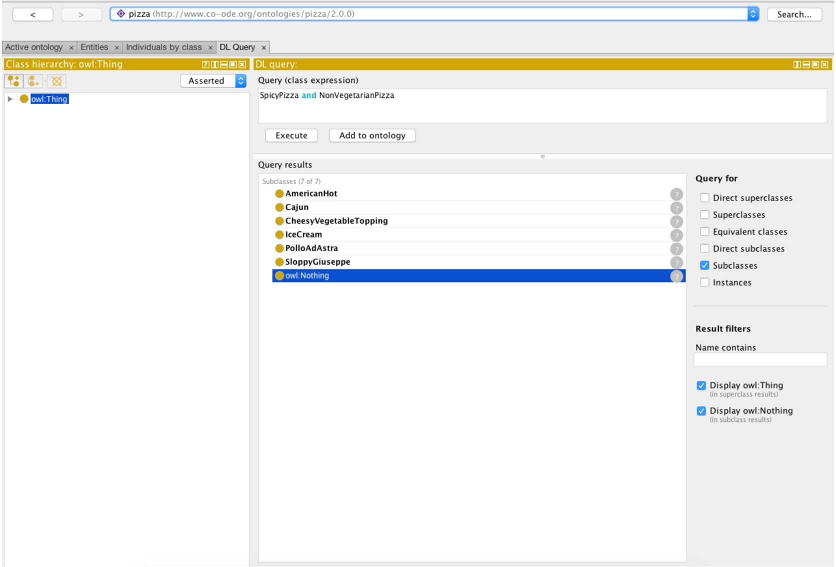
Hands-on Session with Protégé



Hands-on Session with Protégé



Hands-on Session with Protégé



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:

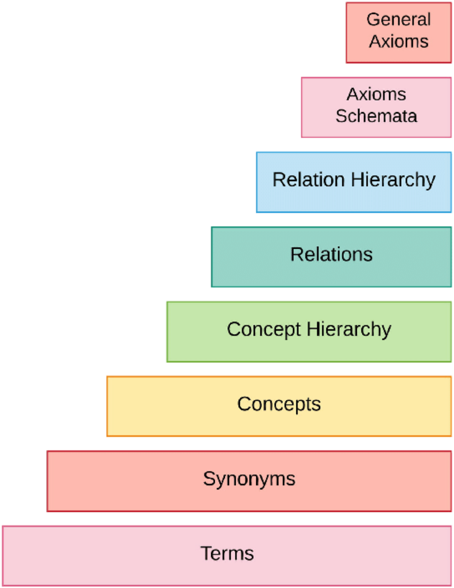
- satisfiability checks;
- entailment computation;
- module extraction;
- ontology alignment.

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 $\sqsubseteq \bot$

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 $\sqsubseteq \bot$
- 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-occurence 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:

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

Positive examples:

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

Negative Examples:

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

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 occuring 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.