The Web Ontology Language (OWL)

Ernesto Jiménez-Ruiz

Lecturer in Artificial Intelligence

Before we start...

Pizza party: Stage 3 and PG students

– **When:** On Wednesday 21st Feb 13:00-14:30.

– **Where:** Lovelace space (College building).

Drop-in hours

- **Term 2**
  - Tuesday 2-3pm (online)
  - Thursdays 2-**3**pm (on-campus). Only up to 3pm today.
- Additional (online) drop-ins can be arranged via email.

Where are we? Module organization.

✓ Introduction: Becoming a knowledge scientist.

✓ RDF-based knowledge graphs.

3. **OWL ontology language. Focus on modelling.** (Today)
4. SPARQL 1.0 Query Language.
5. From tabular data to KG.
6. RDFS Semantics and OWL 2 profiles.
7. Ontology Alignment.
8. Ontology (KG) Embeddings and Machine Learning.
9. SPARQL 1.1 and Graph Database solutions.
10. (Large) Language Models and KGs. (Seminar)

RDF in a nutshell

RDF-based (Knowledge) Graphs

- Resource Description Framework (RDF)
- A standardised data model based on the **directed edge-labelled graph** model.
  - **Nodes**: Internationalised Resource Identifiers (IRIs), literals, and blank nodes (nodes without identifier).
  - **Edges**: IRIs
- **W3C recommendation**.
- Conceptual **modelling of resources**.
- In this module we will study **RDF-based (Knowledge) Graphs**

RDF triples (i)

– RDF-based KGs are composed by *triples* (aka statements or facts)

– A triple consists of *subject*, *predicate*, and *object*

	s	p	o
• IRI/URI references may occur in all positions	✔	✔	✔
• Literals may only occur in object position	✘	✘	✔
• Blank nodes can not occur in predicate position	✔	✘	✔

RDF triples (i)

– RDF-based KGs are composed by *triples* (aka statements or facts)

– A triple consists of *subject*, *predicate*, and *object*

	s	p	o
• IRI/URI references may occur in all positions	✔	✔	✔
• Literals may only occur in object position	✘	✘	✔
• Blank nodes can not occur in predicate position	✔	✘	✔

– Relationships (*i.e.*, edges) are made explicit and are first-class citizens:
  - The predicate is an element in the triple with an IRI.
  - There are also triples describing predicates.

RDF Vocabularies

- Families of related notions are grouped into *vocabularies*.
- Some important, well-known namespaces—and prefixes:

  Modelling vocabulary:

  `rdf:` `<http://www.w3.org/1999/02/22-rdf-syntax-ns#>` – RDF

  `rdfs:` `<http://www.w3.org/2000/01/rdf-schema#>` – RDF Schema

  `owl:` `<http://www.w3.org/2002/07/owl#>` – OWL

  Support vocabularies:

  `dcterms:` `<http://purl.org/dc/terms/>` – Dublin Core

  `bfo:` `<http://purl.obolibrary.org/obo/bfo.owl#>` – Basic Formal Ontology

  `dbo:` `<http://dbpedia.org/ontology/>` – DBPedia Ontology

  `dbr:` `<http://dbpedia.org/resource/>` – DBPedia Resource

RDF Vocabularies

– Families of related notions are grouped into *vocabularies*.

– Some important, well-known namespaces—and prefixes:

Modelling vocabulary:

`rdf:` `<http://www.w3.org/1999/02/22-rdf-syntax-ns#>` – RDF
`rdfs:` `<http://www.w3.org/2000/01/rdf-schema#>` – RDF Schema
`owl:` `<http://www.w3.org/2002/07/owl#>` – OWL

Support vocabularies:

`dcterms:` `<http://purl.org/dc/terms/>` – Dublin Core
`bfo:` `<http://purl.obolibrary.org/obo/bfo.owl#>` – Basic Formal Ontology
`dbo:` `<http://dbpedia.org/ontology/>` – DBPedia Ontology
`dbr:` `<http://dbpedia.org/resource/>` – DBPedia Resource

– Note that the prefix is not standardised.

Example vocabularies: RDF, RDFS

RDF: describing RDF graphs.

- `rdf:Statement`
- `rdf:subject`,
  `rdf:predicate`,
  `rdf:object`
- `rdf:type`

RDFS: describing RDF vocabularies.

- `rdfs:Class`
- `rdfs:subClassOf`,
  `rdfs:subPropertyOf`
- `rdfs:domain`,
  `rdfs:range`
- `rdfs:label`

Examples:

```
dbr:London rdf:type dbo:City .
dbr:London rdfs:label "London"@en .
dbo:City rdfs:subClassOf dbo:Place .
```

Example vocabularies: OWL

OWL: describing ontologies

- `owl:inverseOf`
- `owl:equivalentClass`
- `owl:disjointWith`
- `owl:sameAs`

Examples:

```
dbr:London owl:sameAs ex:London .
dbo:locationOf owl:inverseOf dbo:isLocatedIn .
dbo:City owl:disjointWith dbo:Person .
dbo:City owl:equivalentClass ex:City .
```

Example RDF-based (Knowledge) Graph

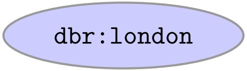
London is a city in England called Londres in Spanish

```
dbr:london rdf:type dbo:City .
dbr:london dbo:locationCountry dbr:england .
dbr:london rdfs:label "Londres"@es .
```

Example RDF-based (Knowledge) Graph

London is a city in England called Londres in Spanish

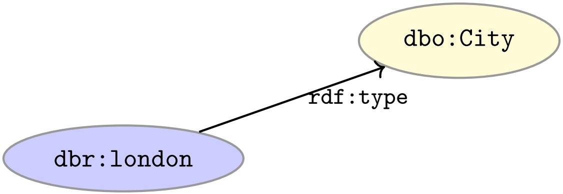
```
dbr:london rdf:type dbo:City .
dbr:london dbo:locationCountry dbr:england .
dbr:london rdfs:label "Londres"@es .
```



Example RDF-based (Knowledge) Graph

London is a city in England called Londres in Spanish

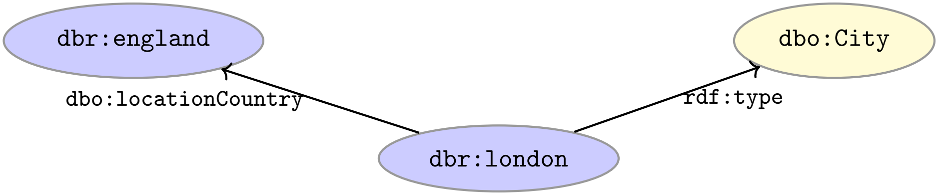
```
dbr:london rdf:type dbo:City .
dbr:london dbo:locationCountry dbr:england .
dbr:london rdfs:label "Londres"@es .
```



Example RDF-based (Knowledge) Graph

London is a city in England called Londres in Spanish

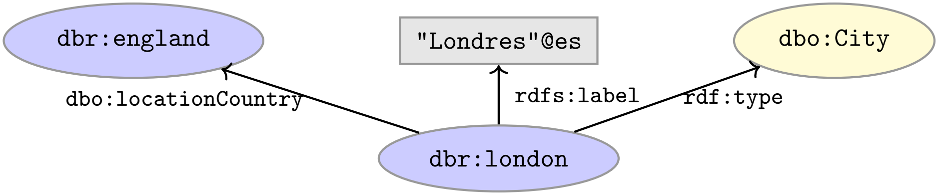
```
dbr:london rdf:type dbo:City .
dbr:london dbo:locationCountry dbr:england .
dbr:london rdfs:label "Londres"@es .
```



Example RDF-based (Knowledge) Graph

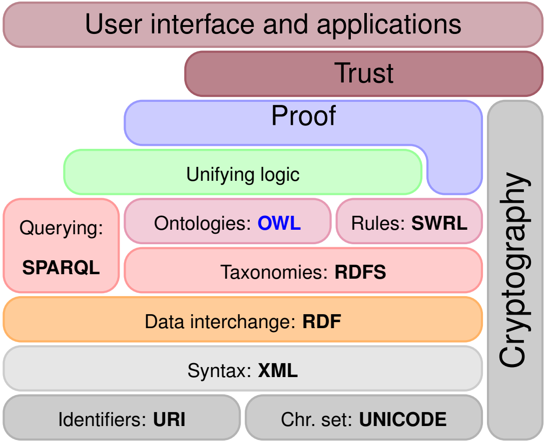
London is a city in England called Londres in Spanish

```
dbr:london rdf:type dbo:City .
dbr:london dbo:locationCountry dbr:england .
dbr:london rdfs:label "Londres"@es .
```



The Web Ontology Language (OWL)

Semantic Web Technology Stack



OWL

- Acronym for ***The Web Ontology Language***.
- A **W3C recommendation**:
  - OWL 1 (2004): `http://www.w3.org/TR/owl-ref/`
  - OWL 2 (2009): `https://www.w3.org/TR/owl2-overview/`

OWL

- Acronym for ***The Web Ontology Language***.
- A **W3C recommendation**:
  - OWL 1 (2004): `http://www.w3.org/TR/owl-ref/`
  - OWL 2 (2009): `https://www.w3.org/TR/owl2-overview/`
- Built on **Description Logics (DL)**.
  - OWL 1: $\mathcal{SHOIN}$(D)
  - OWL 2: $\mathcal{SROIQ}$(D)
- Combines **DL expressiveness with RDF technology**
  - **OWL-layered RDF-based knowledge graphs**

Description Logics and OWL

– **Origin:** semantic networks and other graph-based models and the attempt to formalise them with First Order Logic (FOL).

Description Logics and OWL

- **Origin:** semantic networks and other graph-based models and the attempt to formalise them with First Order Logic (FOL).
- Core reasoning problems in **FOL** are **undecidable**.
- **Trade-off** between **expressiveness** and computational properties

Description Logics and OWL

- **Origin:** semantic networks and other graph-based models and the attempt to formalise them with First Order Logic (FOL).
- Core reasoning problems in **FOL** are **undecidable**.
- **Trade-off** between **expressiveness** and computational properties
- **Description Logics** (DL):
  - Family of knowledge representation languages
  - Decidable subset of FOL
  - Original called: *Terminological language* or *concept language*
  - OWL is based on DL

OWL 1, OWL 2 (profiles) and RDFS

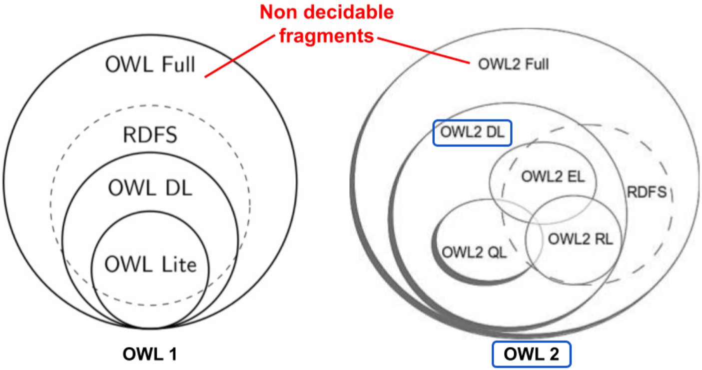


Image adapted from Olivier Cure and Guillaume Blin. RDF Database Systems (Chapter 3). 2015. Elsevier.

Modelling with OWL 2: What is an Ontology?

Ontologies (information sciences)

- Core idea of **knowledge graphs** is the **enhancement of the graph data model** with...
  - “...a **formal specifications** of a shared domain conceptualization”
  - “...an **abstract symbolic representations** of a domain expressed in a formal language”

Thomas R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. 1993
Pim Borst, Hans Akkermans, and Jan Top. Engineering ontologies. 1999.

Ontologies as domain models (i)

- A model is a **simplified** (abstract) **representation** of certain aspects of the real world.

- Models help people **communicate**.

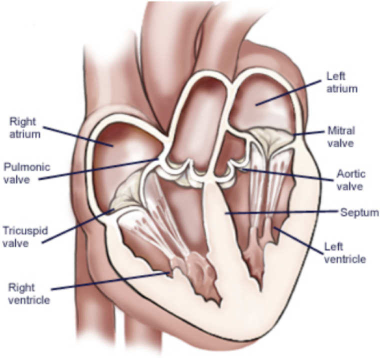
- Models explain and **make predictions**.

- Models **mediate** among multiple viewpoints.

Dean Allemang, James Hendler. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL.

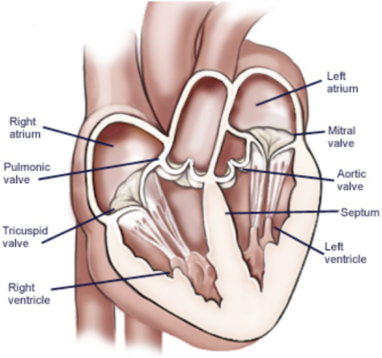
Ontologies as domain models (ii)

- include **vocabulary** relevant to a domain (*e.g.*, with RDF and IRIs)
- specify meaning (**semantics**) of terms (*e.g.*, with OWL)
  - Heart is a muscular organ that is part of the circulatory system



Ontologies as domain models (ii)

- include **vocabulary** relevant to a domain (*e.g.*, with RDF and IRIs)
- specify meaning (**semantics**) of terms (*e.g.*, with OWL)
  - Heart is a muscular organ that is part of the circulatory system
- are **formalised** using a suitable logic language (*e.g.*, with OWL)
  - Heart SUBCLASSOF MuscularOrgan AND (isPartOf SOME CirculatorySystem)



Before Modelling with OWL 2: Set Theory

Sets

$$\{\cdots\}$$

- A set is a mathematical object:

$$\{\text{'a'}, 1, \triangle\}$$

  - Contains 'a', 1, and $\triangle$, and nothing else.
- There is no order between elements

$$\{1, \triangle\} = \{\triangle, 1\}$$

- Nothing can be in a set several times

$$\{1, \triangle, \triangle\} = \{1, \triangle\}$$

- Sets with different elements are different:

$$\{1, 2\} \neq \{2, 3\}$$

Sets: Element of-relation

- $\in$ indicates that something is element of a set:

$$1 \in \{\text{‘a’}, 1, \triangle\}$$

$$\text{‘b’} \notin \{\text{‘a’}, 1, \triangle\}$$

$$\in$$

Sets: Element of-relation

- $\in$ indicates that something is element of a set:

$$1 \in \{\text{‘a’}, 1, \triangle\}$$

$$\text{‘b’} \notin \{\text{‘a’}, 1, \triangle\}$$

$$\in$$

- The set $P_{in3067/inm713}$ of people in the module
  - `city:ernesto` $\in P_{in3067/inm713}$
  - `dbr:Johnny_Depp` $\notin P_{in3067/inm713}$

The Empty Set

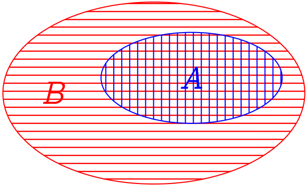
- A set that has no elements.
- This is called the *empty set*
- Notation: $\emptyset$ or $\{\}$
- $x \notin \emptyset$, for any x

$$\emptyset$$

Subsets

- Let A and B be sets
- *if* every element of A is also in B
- *then* A is called a *subset* of B

$$A \subseteq B$$

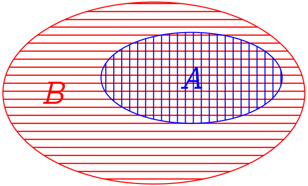


$$\subseteq$$

Subsets

- Let A and B be sets
- *if* every element of A is also in B
- *then* A is called a *subset* of B

$$A \subseteq B$$



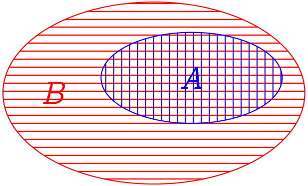
$$\subseteq$$

- Examples
  - $\{\texttt{city:ernesto, city:pravija}\} \subseteq P_{in3067/inm713}$
  - $\{1,3\} \not\subseteq \{1,2\}$
  - $\{1,3\} \subseteq \mathbb{N}$
  - $\emptyset \subseteq A$ for any set A

Subsets

- Let A and B be sets
- *if* every element of A is also in B
- *then* A is called a *subset* of B

$$A \subseteq B$$



$\subseteq$

- Examples
  - $\{\texttt{city:ernesto},\ \texttt{city:pravija}\} \subseteq P_{in3067/inm713}$
  - $\{1,3\} \not\subseteq \{1,2\}$
  - $\{1,3\} \subseteq \mathbb{N}$
  - $\emptyset \subseteq A$ for any set A
- $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$

Intuition: Classes/concepts as Sets

KGs	Set Theory
A `rdf:type owl:Class`	A is a set of resources
x `rdf:type` A	$x \in A$
A `rdfs:subClassOf` B	$A \subseteq B$
`:Person rdf:type owl:Class`	`:Person` is a set of resources
`:ernesto rdf:type :Person`	`:ernesto` $\in$ `:Person`
`:Person rdfs:subClassOf :Animal`	`:Person` $\subseteq$ `:Animal`

Pairs

- A pair is an *ordered* collection of two objects

$$\langle x, y \rangle \qquad\qquad \Big\langle \cdot \;\cdot\; \cdot \Big\rangle$$

- Equal if components are equal:

$$\langle a, b \rangle = \langle x, y \rangle \quad \text{if and only if} \quad a = x \quad \text{and} \quad b = y$$

Pairs

– A pair is an *ordered* collection of two objects

$$\langle x, y \rangle \qquad\qquad \Big\langle \cdot\ \cdot\ \cdot \Big\rangle$$

– Equal if components are equal:

$$\langle a, b \rangle = \langle x, y \rangle \quad \text{if and only if} \quad a = x \quad \text{and} \quad b = y$$

– Order matters:

$$\langle 1, \text{‘a’} \rangle \neq \langle \text{‘a’}, 1 \rangle$$

– An object can be twice in a pair:

$$\langle 1, 1 \rangle$$

The Cross Product

- Let A and B be sets.
- Construct the set of all pairs $\langle a, b \rangle$ with $a \in A$ and $b \in B$.
- This is called the *cross product* of A and B, written

$$A \times B \qquad\qquad \times$$

The Cross Product

- Let A and B be sets.
- Construct the set of all pairs $\langle a, b \rangle$ with $a \in A$ and $b \in B$.
- This is called the *cross product* of A and B, written

$$A \times B$$

$\times$

- Example:
  - $A = \{1, 2, 3\}$, $B = \{\text{'a'}, \text{'b'}\}$.
  - $A \times B = \{ \langle 1, \text{'a'} \rangle, \langle 2, \text{'a'} \rangle, \langle 3, \text{'a'} \rangle, \langle 1, \text{'b'} \rangle, \langle 2, \text{'b'} \rangle, \langle 3, \text{'b'} \rangle \}$

Relations

- A *relation* R between two sets A and B is…
- …a set of pairs $\langle a, b\rangle \in A \times B$

$$R \subseteq A \times B$$

- R connects elements of A with elements of B. For example:

$$teachesAt \subseteq Academic \times University$$

- We often write '$a\ R\ b$' to say that $\langle a, b\rangle \in R$. For example:

$$\langle ernesto, city\rangle \in teachesAt$$

- A relation R *on* a single set A is a relation between A and A:

$$R \subseteq A \times A = A^2$$

Example: Family Relations

- Consider the set $A = \{\text{Homer}, \text{Marge}, \text{Bart}, \text{Lisa}, \text{Maggie}\}$.
- Consider a relation P on A such that

$$x \; P \; y \quad \text{iff} \quad x \textbf{ is parent of } y$$

- As a **set of pairs**:

$$\begin{array}{lllll} P = \{ & \langle \text{Homer}, \text{Bart} \rangle, & \langle \text{Homer}, \text{Lisa} \rangle, & \langle \text{Homer}, \text{Maggie} \rangle, & \\ & \langle \text{Marge}, \text{Bart} \rangle, & \langle \text{Marge}, \text{Lisa} \rangle, & \langle \text{Marge}, \text{Maggie} \rangle & \} \subseteq A^2 \end{array}$$

- For instance:

$$\langle \text{Homer}, \text{Bart} \rangle \in P \qquad \langle \text{Marge}, \text{Maggie} \rangle \in P \qquad \langle \text{Bart}, \text{Homer} \rangle \notin P$$

Modelling with OWL 2: Introduction

Description Logics and Interpretations

- **Interpretations** ($\mathcal{I}$) might be conceived as potential "realities".
- They can be seen as a **function** from abstract representation to concrete elements in set theory.
- Interpretations **assign values** to elements and may be the **model** of a graph or ontology (*i.e.*, $\mathcal{I}$ entails all its elements).
- For example *(very small interpretation of the world)*:
  - $\texttt{dbp:london}^{\mathcal{I}} = \texttt{dbp:london}^{\mathcal{I}}$
  - $\texttt{dbo:City}^{\mathcal{I}} = \{\texttt{dbp:london}^{\mathcal{I}}\}$
  - $\texttt{dbo:Country}^{\mathcal{I}} = \{\texttt{dbp:england}^{\mathcal{I}}\}$
  - $\texttt{dbo:PopulatedPlace}^{\mathcal{I}} = \{\texttt{dbp:london}^{\mathcal{I}}, \texttt{dbp:england}^{\mathcal{I}}\}$
  - $\texttt{dbo:isLocatedIn}^{\mathcal{I}} = \{\langle \texttt{dbp:london}^{\mathcal{I}},\ \texttt{dbp:england}^{\mathcal{I}} \rangle\}$

Description Logics Syntax (first steps)

KG Triples	DL Syntax	Semantics
`:london :location :england .`	$location(london, england)$	$\langle london^{\mathcal{I}}, england^{\mathcal{I}} \rangle \in location^{\mathcal{I}}$
`:london rdf:type :City .`	$City(london)$	$london^{\mathcal{I}} \in City^{\mathcal{I}}$
`:City rdfs:subClassOf :Place .`	$City \sqsubseteq Place$	$City^{\mathcal{I}} \subseteq Place^{\mathcal{I}}$
`:capitalOf rdfs:subPropOf :location .`	$capitalOf \sqsubseteq location$	$capitalOf^{\mathcal{I}} \subseteq location^{\mathcal{I}}$

(*) Not all OWL 2 axioms have a 1-to-1 triple representation.

OWL 2 entities: classes and individuals

`owl:Class`: to represent classes (*i.e.*, set of individuals).

- Atomic (with a IRI) `:Person rdf:type owl:Class`
- Complex (built from other entities)

OWL 2 entities: classes and individuals

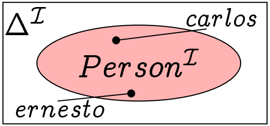
`owl:Class`: to represent classes (*i.e.*, set of individuals).

- Atomic (with a IRI) `:Person rdf:type owl:Class`
- Complex (built from other entities)

`owl:NamedIndividual`, to represent individuals

```
:ernesto rdf:type owl:NamedIndividual
:ernesto rdf:type :Person
```

$:ernesto^{\mathcal{I}} \in :Person^{\mathcal{I}}$



OWL 2 entities: Top and Bottom classes

`owl:Thing`

- $\top$ (in DL syntax)
- Class containing **all individuals**.
- Its interpretation is $\Delta^{\mathcal{I}}$.
- For every `owl:Class` C, C is a subclass of `owl:Thing`.

OWL 2 entities: Top and Bottom classes

`owl:Thing`

- $\top$ (in DL syntax)
- Class containing **all individuals**.
- Its interpretation is $\Delta^{\mathcal{I}}$.
- For every `owl:Class` C, C is a subclass of `owl:Thing`.

`owl:Nothing`

- $\bot$ (in DL syntax)
- **empty class** containing no individuals.
- Its interpretation is the empty set $\emptyset$
- For every `owl:Class` C, `owl:Nothing` is a subclass of C

OWL 2 entities: properties

OWL Properties (instead of `rdf:Property`):

- `owl:DatatypeProperty`. Targets data values.

  `:hasName rdf:type owl:DatatypeProperty.`

  Universal data property: $\mathcal{D}^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Lambda$ (Λ set of all literal values)
- `owl:ObjectProperty`. Targets individuals.

  `:teaches rdf:type owl:ObjectProperty`

  Universal object property: $U^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
- `owl:AnnotationProperty`. No logical implication.

  `rdfs:label rdf:type owl:AnnotationProperty.`

The set of classes, named individuals, annotation, data and object properties are **mutually disjoint**.

Open World Assumptions

Closed World Assumption (**CWA**)

- Complete knowledge.
- Any statement that is not known to be true is false. ($*$)
- Typical semantics for **database systems**.

Open World Assumptions

Closed World Assumption (**CWA**)

- Complete knowledge.
- Any statement that is not known to be true is false. ($*$)
- Typical semantics for **database systems**.

Open World Assumption (**OWA**)

- Potential incomplete knowledge.
- ($*$) does not hold.
- Typical semantics for **logic-based systems** (including OWL).

Name Assumptions

Unique Name Assumption (**UNA**)

- Different names **always** denote different things.
  - E.g., $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.
  - common in relational databases.

Name Assumptions

Unique Name Assumption (**UNA**)

- Different names **always** denote different things.
  - E.g., $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.
  - common in relational databases.

Non-unique Name Assumption (**NUNA**)

- Different names **need not** denote different things. **As in OWL**.
  - $\texttt{dbpedia:Person}^{\mathcal{I}} = \texttt{foaf:Person}^{\mathcal{I}}$.
  - $\texttt{wikidata:ernesto}^{\mathcal{I}}$ = $\texttt{city:ernesto}^{\mathcal{I}}$

Name Assumptions

Unique Name Assumption (**UNA**)

- Different names **always** denote different things.
  - E.g., $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.
  - common in relational databases.

Non-unique Name Assumption (**NUNA**)

- Different names **need not** denote different things. **As in OWL**.
  - $\texttt{dbpedia:Person}^{\mathcal{I}} = \texttt{foaf:Person}^{\mathcal{I}}$.
  - $\texttt{wikidata:ernesto}^{\mathcal{I}}$ = $\texttt{city:ernesto}^{\mathcal{I}}$

Equal names (*e.g.*, URIs) always denote the same “thing”.

- E.g., cannot have $\texttt{city:ernesto}^{\mathcal{I}} \neq \texttt{city:ernesto}^{\mathcal{I}}$.

Modelling with OWL 2: Axioms and Class Constructs

OWL 2 Axioms

- OWL 2 ontologies are composed by **axioms**.
- **Not all axioms have a 1-to-1 triple representation.**
- Historically, axioms are put in **boxes**.

OWL 2 Axioms

- OWL 2 ontologies are composed by **axioms**.
- **Not all axioms have a 1-to-1 triple representation.**
- Historically, axioms are put in **boxes**.
- **The TBox** (terminological knowledge)
  - is typically **independent of** any actual **instance data**.
  - Class inclusion $C \sqsubseteq D$, equivalence $C \equiv D$
  - The set of property axioms are also referred to as **RBox.**

OWL 2 Axioms

– OWL 2 ontologies are composed by **axioms**.

– **Not all axioms have a 1-to-1 triple representation.**

– Historically, axioms are put in **boxes**.

– **The TBox** (terminological knowledge)
  – is typically **independent of** any actual **instance data**.
  – Class inclusion $C \sqsubseteq D$, equivalence $C \equiv D$
  – The set of property axioms are also referred to as **RBox.**

– **The ABox** (assertional knowledge)
  – contains facts about concrete instances (basically as in RDF)

OWL 2 TBox Axioms

- The TBox (excluding the RBox) is composed by:
  - **Subsumption axioms:** $C \sqsubseteq D$ ($C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$)
  - **Equivalence axioms:** $C \equiv D$ ($C^{\mathcal{I}} = D^{\mathcal{I}}$)

OWL 2 TBox Axioms

- The TBox (excluding the RBox) is composed by:
  - **Subsumption axioms:** $C \sqsubseteq D$ ($C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$)
  - **Equivalence axioms:** $C \equiv D$ ($C^{\mathcal{I}} = D^{\mathcal{I}}$)
- C and D can be **named concepts** (*e.g.*, `dbo:City`) or **complex concepts** built from others.
  - **Negation**: $\neg E$ (not)
  - **Intersection**: $E \sqcap F$ (and)
  - **Union**: $E \sqcup F$ (or)
  - **Property restrictions**: $\exists R.E$ (some), $\forall R.E$ (only)

OWL 2 TBox Axioms

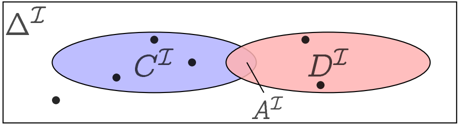
- The TBox (excluding the RBox) is composed by:
  - **Subsumption axioms:** $C \sqsubseteq D$ ($C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$)
  - **Equivalence axioms:** $C \equiv D$ ($C^{\mathcal{I}} = D^{\mathcal{I}}$)

- C and D can be **named concepts** (*e.g.*, `dbo:City`) or **complex concepts** built from others.
  - **Negation**: $\neg E$ (not)
  - **Intersection**: $E \sqcap F$ (and)
  - **Union**: $E \sqcup F$ (or)
  - **Property restrictions**: $\exists R.E$ (some), $\forall R.E$ (only)

- We will focus on the cases where the **LHS concept is atomic**. *e.g.*, `dbo:City` $\sqsubseteq$ `dbo:Place` $\sqcap$ ...

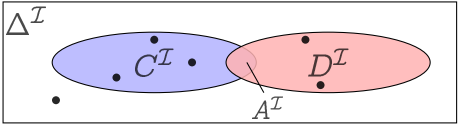
Union and Intersection

- $A \sqsubseteq C \sqcap D$. "A is both C and D."
  - *e.g.*, $\mathtt{DryRedWine} \sqsubseteq \mathtt{DryWine} \sqcap \mathtt{RedWine}$.

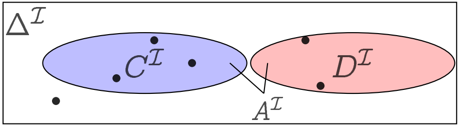


Union and Intersection

- $A \sqsubseteq C \sqcap D$. "A is both C and D."
  - *e.g.*, $\texttt{DryRedWine} \sqsubseteq \texttt{DryWine} \sqcap \texttt{RedWine}$.

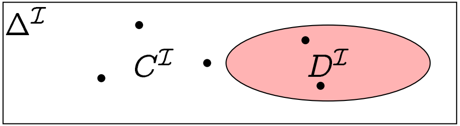


- $A \sqsubseteq C \sqcup D$. "A is C or D."
  - *e.g.*, $\texttt{Wine} \sqsubseteq \texttt{BadWine} \sqcup \texttt{GoodWine}$.



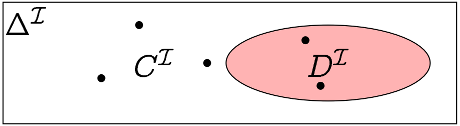
Negation and Disjointness

– $C \sqsubseteq \neg D$: “C is not D.”
  – *e.g.*, $\texttt{VegetarianTopping} \sqsubseteq \neg\texttt{MeatTopping}$.

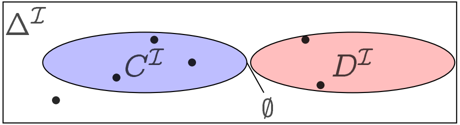


Negation and Disjointness

- $C \sqsubseteq \neg D$: "C is not D."
  - *e.g.*, $\texttt{VegetarianTopping} \sqsubseteq \neg\texttt{MeatTopping}$.



- $C \sqcap D \sqsubseteq \bot$: "Nothing is both a C and a D."
  - Equivalent to $C \sqsubseteq \neg D$ (and $D \sqsubseteq \neg C$).
  - *e.g.*, $\texttt{VegetarianTopping} \sqcap \texttt{MeatTopping} \sqsubseteq \bot$.

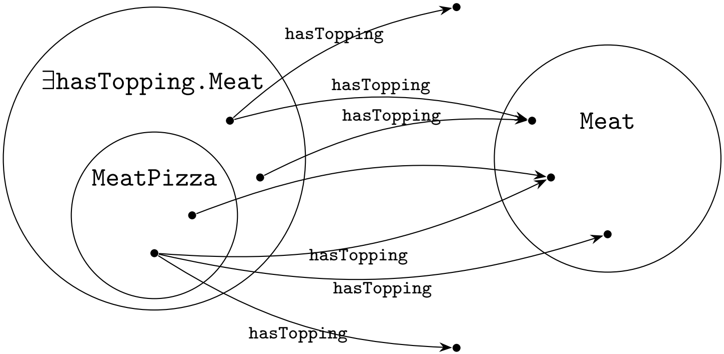


Existential Restrictions

- $A \sqsubseteq \exists R.C$: "$A$ is R-related to (at least) one C."
- $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid$ there is a b where $\langle a, b\rangle \in R^{\mathcal{I}}$ and $b \in C^{\mathcal{I}}\}$

Existential Restrictions

- $A \sqsubseteq \exists R.C$: "$A$ is R-related to (at least) one C."
- $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid$ there is a b where $\langle a, b\rangle \in R^{\mathcal{I}}$ and $b \in C^{\mathcal{I}}\}$
- *e.g.*, $\texttt{MeatPizza} \sqsubseteq \exists\texttt{hasTopping.Meat}$



Existential Restrictions and `rdfs:domain`

Local scope for R:

- $\texttt{Pizza} \sqsubseteq \exists \texttt{hasTopping}.\texttt{Topping}$

Domain: (global scope for R)

- If R has the *domain* C: (`R rdfs:domain C`)
- then anything 'using' R is in C.
- Domain can also be expressed as: $\exists R.\top \sqsubseteq C$
- $\exists \texttt{hasTopping}.\top \sqsubseteq \texttt{Pizza}$

Existential Restrictions and `rdfs:domain`

Local scope for R:

- $\texttt{Pizza} \sqsubseteq \exists \texttt{hasTopping.Topping}$

Domain: (global scope for R)

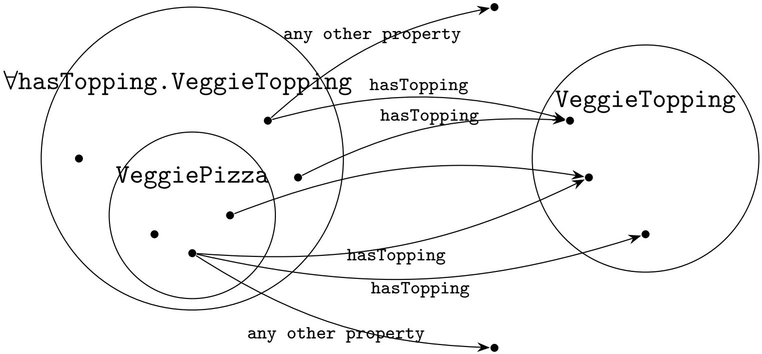
- If R has the *domain* C: (`R rdfs:domain C`)
- then anything 'using' R is in C.
- Domain can also be expressed as: $\exists R.\top \sqsubseteq C$
- $\exists \texttt{hasTopping}.\top \sqsubseteq \texttt{Pizza}$
- Examples:
  - `:pizza1 :hasTopping :meat1`
  - `:ice-cream1 :hasTopping :choco-chips1`

Universal Restrictions

- $A \sqsubseteq \forall R.C$: A has R-relationships to C's only.
- $(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid$ for all b, if $\langle a, b\rangle \in R^{\mathcal{I}}$ then $b \in C^{\mathcal{I}}\}$

Universal Restrictions

- $A \sqsubseteq \forall R.C$: A has R-relationships to C's only.
- $(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \text{for all } b, \text{ if } \langle a, b \rangle \in R^{\mathcal{I}} \text{ then } b \in C^{\mathcal{I}}\}$
- *e.g.*, $\texttt{VeggiePizza} \sqsubseteq \forall\texttt{hasTopping.VeggieTopping}$



Universal Restrictions and `rdfs:range`

Local scope for R:

- $\mathtt{VeggiePizza} \sqsubseteq \forall \mathtt{hasTopping}.\mathtt{VeggieTopping}$.

Range: (global scope for R)

- If role R has the *range* C: (`R rdfs:range C`)
- then anything one can reach by R is in C,
- Range can also be expressed as $\top \sqsubseteq \forall R.C$.
- $\top \sqsubseteq \forall \mathtt{hasTopping}.\mathtt{VeggieTopping}$.

Universal Restrictions and `rdfs:range`

Local scope for R:

- $\texttt{VeggiePizza} \sqsubseteq \forall \texttt{hasTopping}.\texttt{VeggieTopping}$.

Range: (global scope for R)

- If role R has the *range* C: (`R rdfs:range C`)
- then anything one can reach by R is in C,
- Range can also be expressed as $\top \sqsubseteq \forall R.C$.
- $\top \sqsubseteq \forall \texttt{hasTopping}.\texttt{VeggieTopping}$.
- Example:
  - `:pizza1 :hasTopping :meat1` (is `meat1` a `VeggieTopping`?)

Universal Restrictions and `rdfs:range`

Local scope for R:

- $\texttt{VeggiePizza} \sqsubseteq \forall \texttt{hasTopping}.\texttt{VeggieTopping}$.

Range: (global scope for R)

- If role R has the *range* C: (`R rdfs:range C`)
- then anything one can reach by R is in C,
- Range can also be expressed as $\top \sqsubseteq \forall R.C$.
- $\cancel{\top \sqsubseteq \forall \texttt{hasTopping}.\texttt{VeggieTopping}}$. $\top \sqsubseteq \forall \texttt{hasTopping}.\texttt{PizzaTopping}$.
- Example:
  - `:pizza1 :hasTopping :meat1`

Cardinality restrictions

- Restricts the number of relations a type of object can have.
- Syntax:
  - $\leq_n R.C$, $\geq_n R.C$, and $=_n R.C$.

Cardinality restrictions

– Restricts the number of relations a type of object can have.

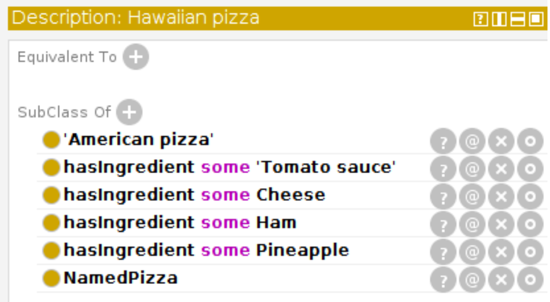
– Syntax:
  – $\leq_n R.C$, $\geq_n R.C$, and $=_n R.C$.

– Axioms read:
  – $A \sqsubseteq \ \square_n R.C$: “An element of A is R-related to n number of C’s.”
    – $\leq$: *at most*
    – $\geq$: *at least*
    – $=$: *exactly*

– *e.g.*, $\texttt{SuperMeatPizza} \sqsubseteq \geq_5 \texttt{hasTopping.Meat}$

Necessary conditions and primitive classes

Hawaiian pizza **implies** having pineapple as ingredient (among others); but not the other way round.

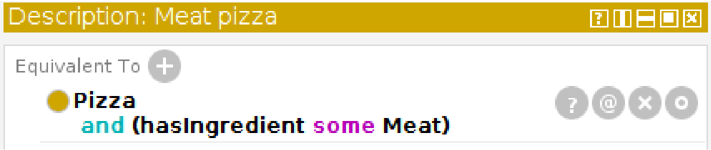


Sufficient conditions and defined classes

Meat pizza **implies** having meat as ingredient (and being pizza).

A pizza with meat as ingredient **implies** being a meat pizza.

Hawaiian pizzas have ham as ingredient and thus they are meat pizzas.

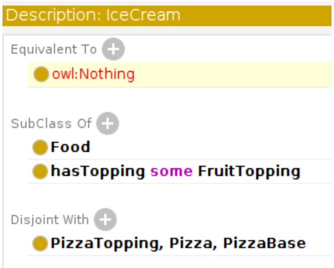


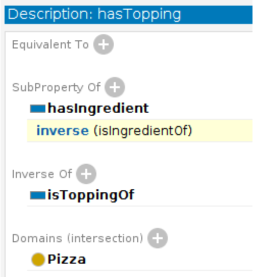
Detecting modelling errors

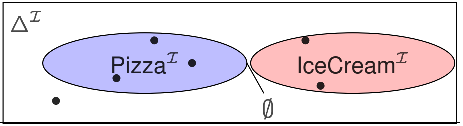
Ice cream **implies** having fruit as topping

Ice cream is **disjoint with** Pizza

The domain of has topping is pizza, that is, having any topping **implies** being a pizza (domain is a type of sufficient condition).







Modeling with OWL 2: RBox

Property axioms

- subsumption:
  $\texttt{hasBrother} \sqsubseteq \texttt{hasSibling}$
- equivalence:
  $\texttt{hasLocation} \equiv \texttt{locatedIn}$
- inverse roles (only object properties):
  $\texttt{hasParent} \equiv \texttt{hasChild}^{-1}$
- role chains (only object properties):
  $\texttt{hasParent} \circ \texttt{hasBrother} \sqsubseteq \texttt{hasUncle}$
  $(R \circ S)^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, c^{\mathcal{I}}\rangle \mid \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}, \langle b^{\mathcal{I}}, c^{\mathcal{I}}\rangle \in S^{\mathcal{I}}\}$

Common characteristics for (object) properties

A relation R over the set $\Delta^{\mathcal{I}}$ ($R \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$) is

Characteristic	Semantics	Example
Reflexive:	if $\langle a, a \rangle \in R$ for all $a \in \Delta^{\mathcal{I}}$	`part_of`
Irreflexive:	if $\langle a, a \rangle \notin R$ for all $a \in X$	`hasParent`

Common characteristics for (object) properties

A relation R over the set $\Delta^{\mathcal{I}}$ ($R \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$) is

Characteristic	**Semantics**	**Example**
Reflexive:	if $\langle a, a \rangle \in R$ for all $a \in \Delta^{\mathcal{I}}$	`part_of`
Irreflexive:	if $\langle a, a \rangle \notin R$ for all $a \in X$	`hasParent`
Symmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \in R$	`hasSibling`
Asymmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \notin R$	`memberOf/hasParent`

Common characteristics for (object) properties

A relation R over the set $\Delta^{\mathcal{I}}$ ($R \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$) is

Characteristic	**Semantics**	**Example**
Reflexive:	if $\langle a, a \rangle \in R$ for all $a \in \Delta^{\mathcal{I}}$	`part_of`
Irreflexive:	if $\langle a, a \rangle \notin R$ for all $a \in X$	`hasParent`
Symmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \in R$	`hasSibling`
Asymmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \notin R$	`memberOf/hasParent`
Transitive:	if $\langle a, b \rangle$, $\langle b, c \rangle \in R$ implies $\langle a, c \rangle \in R$	`locatedIn`

Common characteristics for (object) properties

A relation R over the set $\Delta^{\mathcal{I}}$ ($R \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$) is

Characteristic	Semantics	Example
Reflexive:	if $\langle a, a \rangle \in R$ for all $a \in \Delta^{\mathcal{I}}$	`part_of`
Irreflexive:	if $\langle a, a \rangle \notin R$ for all $a \in X$	`hasParent`
Symmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \in R$	`hasSibling`
Asymmetric:	if $\langle a, b \rangle \in R$ implies $\langle b, a \rangle \notin R$	`memberOf/hasParent`
Transitive:	if $\langle a, b \rangle, \langle b, c \rangle \in R$ implies $\langle a, c \rangle \in R$	`locatedIn`
Functional:	if $\langle a, b \rangle, \langle a, c \rangle \in R$ implies $b = c$	`hasBiologicalMother`
Inverse functional:	if $\langle a, b \rangle, \langle c, b \rangle \in R$ implies $a = c$	`gaveBirthTo`

(*) Functional characteristics can also be applied to data properties.

Datatypes for data properties

- Many predefined datatypes are available in OWL:
  - all common XSD datatypes: `xsd:string`, `xsd:int`, ...
  - a few from RDF: `rdf:PlainLiteral`,
  - and a few of their own: `owl:real` and `owl:rational`.

Datatypes for data properties

- Many predefined datatypes are available in OWL:
  - all common XSD datatypes: `xsd:string`, `xsd:int`, ...
  - a few from RDF: `rdf:PlainLiteral`,
  - and a few of their own: `owl:real` and `owl:rational`.
- Datatypes may be restricted with *constraining facets*, borrowed from XML Schema.
  - `Teenager` is equivalent to:

    $\texttt{Person} \sqcap (\exists \texttt{age.xsd:integer[>= 13, <= 19]})$

Modeling with OWL 2: ABox

ABox: Assertional axioms

Contains:

- Facts about concrete instances $a, b, c, \ldots$
- A set of concept assertions $C(a)$ as in RDF
  DL: `Person(ernesto)`
  Triple `:ernesto rdf:type :Person`

ABox: Assertional axioms

Contains:

- Facts about concrete instances $a, b, c, \ldots$
- A set of concept assertions $C(a)$ as in RDF
  DL: `Person(ernesto)`
  Triple `:ernesto rdf:type :Person`
- Role assertions $R(b, c)$ as in RDF
  DL: `teaches(ernesto, inm713-in3067)`
  Triple: `:ernesto :teaches :inm713-in3067`

ABox: Assertional axioms

Contains:

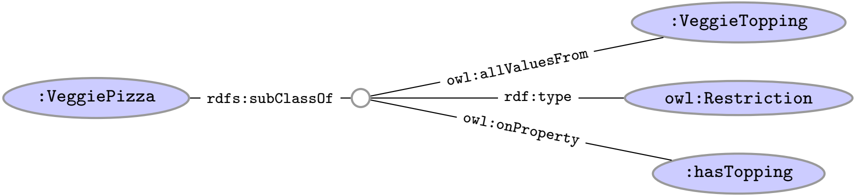
- Facts about concrete instances $a, b, c, \ldots$
- A set of concept assertions $C(a)$ as in RDF
  DL: `Person(ernesto)`
  Triple `:ernesto rdf:type :Person`
- Role assertions $R(b, c)$ as in RDF
  DL: `teaches(ernesto, inm713-in3067)`
  Triple: `:ernesto :teaches :inm713-in3067`
- Equality and non-equality between individuals
  - DL: $\mathtt{ernesto} = \mathtt{ejr}$, $\mathtt{ernesto} \neq \mathtt{aidan}$;
  - Triple (1) `:ernesto owl:sameAs :ejr` (2) `:ernesto owl:differentFrom :aidan`

OWL 2 Syntaxes and Serialization

OWL Syntaxes

– OWL (as RDF) is an abstract construction with several syntaxes.

– $\texttt{VeggiePizza} \sqsubseteq \forall \texttt{hasTopping}.\texttt{VeggieTopping}$



– In Turtle syntax (storage basically as triples):

```
:VeggiePizza rdfs:subClassOf [ rdf:type owl:Restriction ;
                               owl:onProperty :hasTopping ;
                               owl:allValuesFrom :VeggieTopping ] .
```

Manchester OWL Syntax

– Used in Protégé for concept descriptions.

– Correspondence to DL constructs:

DL	Manchester
$C \sqcap D$	C **and** D
$C \sqcup D$	C **or** D
$\neg C$	**not** C
$\forall R.C$	R **only** C
$\exists R.C$	R **some** C
$\leq_n R.D$	R **max** n C
$\geq_n R.D$	R **min** n C
$=_n R.D$	R **exactly** n C

OWL 2 Reasoning

Automated Reasoning (i)

- Formal semantics allows the automatic deduction of **new facts**.
- Also allows us to perform checks that aim to detect the **correctness** of the designed model (*e.g.*, `:dolphin` is a `:Fish`?).

Automated Reasoning (i)

- Formal semantics allows the automatic deduction of **new facts**.
- Also allows us to perform checks that aim to detect the **correctness** of the designed model (*e.g.*, `:dolphin` is a `:Fish`?).
- Possibly in the form of **obvious errors**:
  - `:Mammal` and `:Fish` are disjoint classes.
  - `:dolphin` cannot be an individual (or a subclass) of both `:Mammal` and `:Fish`.

Automated Reasoning (i)

- Formal semantics allows the automatic deduction of **new facts**.
- Also allows us to perform checks that aim to detect the **correctness** of the designed model (*e.g.*, `:dolphin` is a `:Fish`?).
- Possibly in the form of **obvious errors**:
  - `:Mammal` and `:Fish` are disjoint classes.
  - `:dolphin` cannot be an individual (or a subclass) of both `:Mammal` and `:Fish`.
- Extremely valuable for designing **correct** ontologies/KGs, specially when working **collaboratively** and **integrating** various sources.

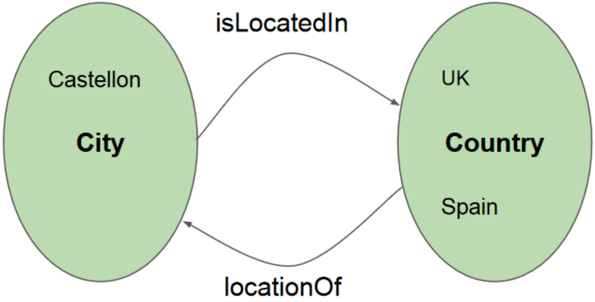
Automated Reasoning (ii)

- More after the reading week: **RDFS semantics and OWL 2 profiles.**
- Today we will use **HermiT reasoner in Protégé** to entail implicit knowledge within the KG.

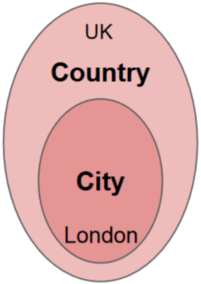
Common modellings mistakes and misunderstandings

Common mistakes: part-of VS subclass-of (i)

City **subClassOf** isLocatedIn **some** Country

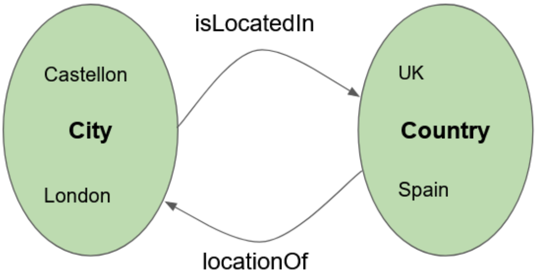


City **subClassOf** Country

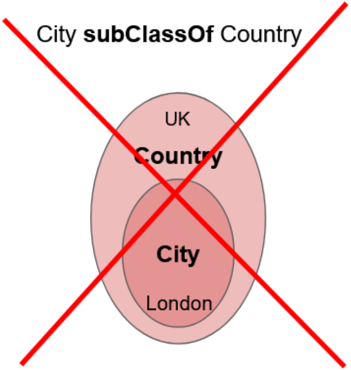


Common mistakes: part-of VS subclass-of (i)

City **subClassOf** isLocatedIn **some** Country

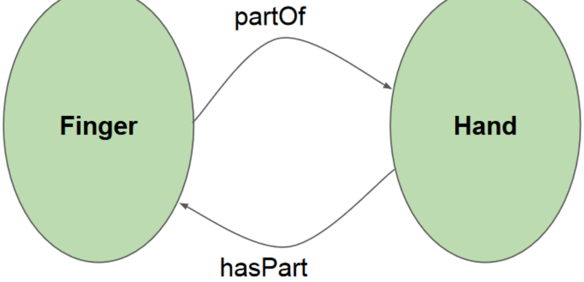


City **subClassOf** Country

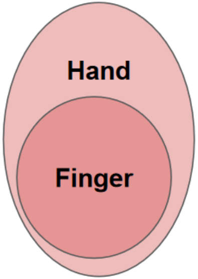


Common mistakes: part-of VS subclass-of (ii)

Finger **subClassOf** partOf **some** Hand

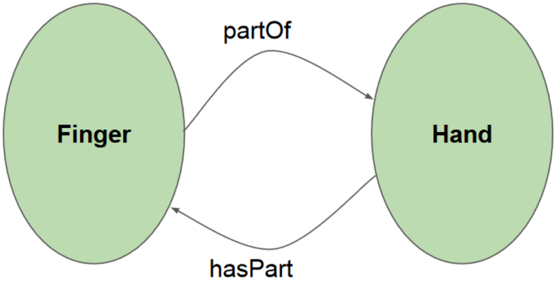


Finger **subClassOf** Hand

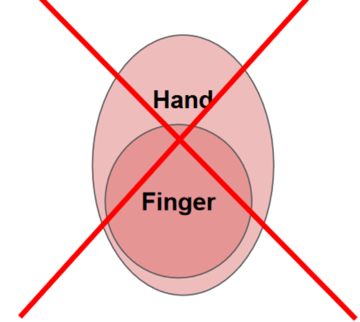


Common mistakes: part-of VS subclass-of (ii)

Finger **subClassOf** partOf **some** Hand

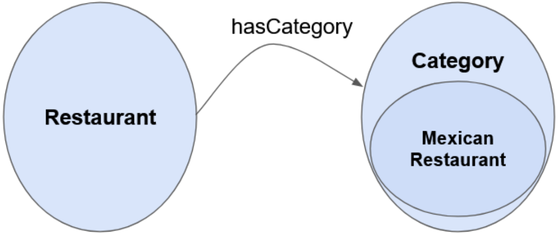


Finger **subClassOf** Hand

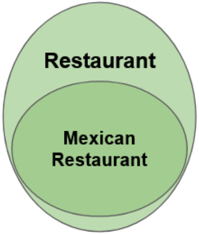


Common mistakes: part-of VS subclass-of (iii)

Restaurant **subClassOf** hasCategory **some** Category

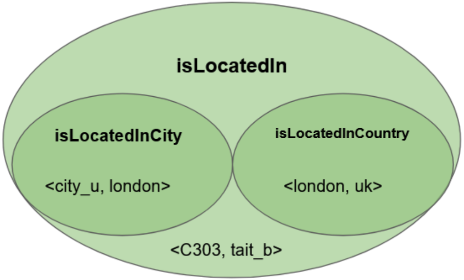


Mexican_Restaurant **subClassOf** Restaurant

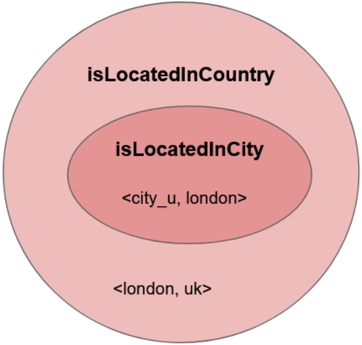


Common mistakes: property hierarchy

isLocatedInCity **subPropertyOf** isLocatedIn
isLocatedInCountry **subPropertyOf** isLocatedIn

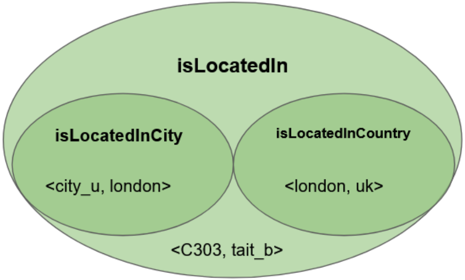


isLocatedInCity **subPropertyOf** isLocatedinCountry

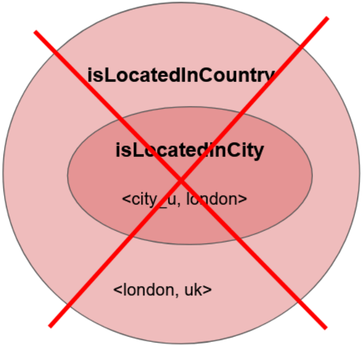


Common mistakes: property hierarchy

isLocatedInCity **subPropertyOf** isLocatedIn
isLocatedInCountry **subPropertyOf** isLocatedIn

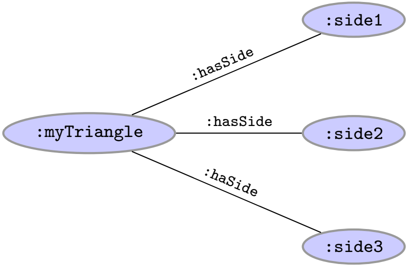


isLocatedInCity **subPropertyOf**
isLocatedinCountry



OWL 2 and Open World Assumption

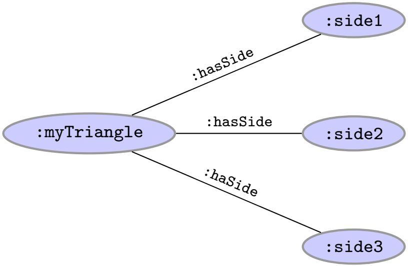
– `:Triangle EquivalentTo :hasSide exactly 3 :Side`



– is `:myTriangle` a `:Triangle`?

OWL 2 and Open World Assumption

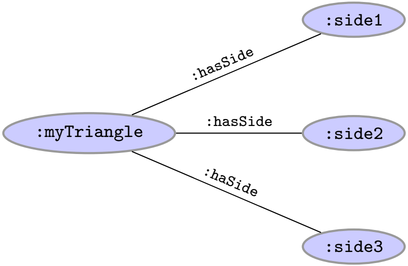
- `:Triangle EquivalentTo :hasSide exactly 3 :Side`



- is `:myTriangle` a `:Triangle`? I don't know because of OWA and NUNA.

OWL 2 and Open World Assumption

– `:Triangle EquivalentTo :hasSide exactly 3 :Side`



– is `:myTriangle` a `:Triangle`? I don't know because of OWA and NUNA.

– Solution: reasoning in OWL complemented with SPARQL queries (in this case with aggregates) → SPARQL 1.1 (not today)

Modelling ontologies outside OWL 2

- Combination of OWL 2 DL axioms leading to an ontology outside OWL 2 DL.
- These combination can easily be done in Protégé.

Michael Schneider et al. *Modeling in OWL 2 without Restrictions*: `https://arxiv.org/pdf/1212.2902.pdf`.

Modelling ontologies outside OWL 2

- Combination of OWL 2 DL axioms leading to an ontology outside OWL 2 DL.
- These combination can easily be done in Protégé.
- Very common (not allowed in OWL 2 DL):
  - cardinality restrictions on transitive properties
  - property chains on functional properties
  - transitive and asymmetric properties

Michael Schneider et al. *Modeling in OWL 2 without Restrictions*: `https://arxiv.org/pdf/1212.2902.pdf`.

Lab Session

Laboratory with OWL

- Modelling ontologies with Protégé.
  - Protégé presents ontologies almost like an OO modelling tool.
  - Pizza OWL tutorial.

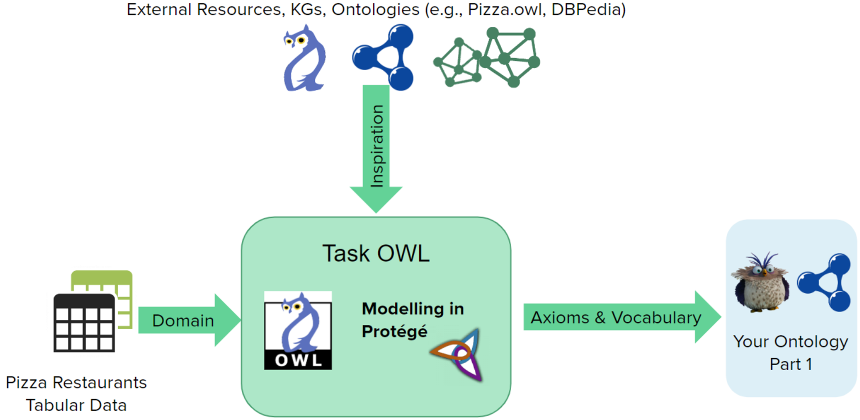
- Short demo (in moodle): Week 2.

Coursework Part 1: Ontology modelling (i)

- Part 1 (20%): **creation of an ontology** that covers the knowledge of a given domain. **Deadline**: Sunday, 3 March 2024, 5:00 PM

- **Work in pairs**, or individually. Please register by February 23.

Coursework Part 1: Ontology modelling (ii)



(*) Ontology CW2 = Model solution $\neq$ Your Ontology Part 1

Acknowledgements

Acknowledgements

– Prof. Martin Giese and others (University of Oslo)
  – `INF4580 - Semantic technologies`
  – `https://www.uio.no/studier/emner/matnat/ifi/INF4580/`

– Dr. Valentina Tamma (University of Liverpool)
  – `Comp 318 - Advanced Web Technologies`
  – `https://cgi.csc.liv.ac.uk/~valli/Comp318.html`