The Web Ontology Language (OWL)

Ernesto Jiménez-Ruiz
Lecturer in Artificial Intelligence

Before we start...

Q&A Reading Week

- **Your duties**: Please bring questions. Ideally: post them in forum in advance.
- **My duty**:
  - Prepare course work material. We will start putting things into the Data Science workflow.
  - Available from 9am-12am.
- Student feedback.

MSc projects

- Some topics available in moodle.
- Deadlines:
  - Supervisor-Student pairing (PAT system) by **April 14**.
  - RMPI project proposal by **April 28**.
- Role of the supervisor.
  - It depends... *(from consultancy to close supervision)*
  - Moderation but **not marking**.

Where are we?

Where are we?

- ✓ RDF-based knowledge graphs.
- ✓ SPARQL 1.0
- ✓ RDFS Semantics and RDF(S)-based knowledge graphs.

Where are we?

- ✓ RDF-based knowledge graphs.
- ✓ SPARQL 1.0
- ✓ RDFS Semantics and RDF(S)-based knowledge graphs.
- – **OWL (2) ontology language (Today)**. Focus on modelling.

Where are we?

- ✓ RDF-based knowledge graphs.
- ✓ SPARQL 1.0
- ✓ RDFS Semantics and RDF(S)-based knowledge graphs.
- – **OWL (2) ontology language (Today)**. Focus on modelling.
- – SPARQL 1.1, OWL 2 profiles and entailment regimes.
- – Application to Data Science.
- – Ontology Alignment.
- – Machine Learning and Knowledge Graphs.

Recap: RDFS

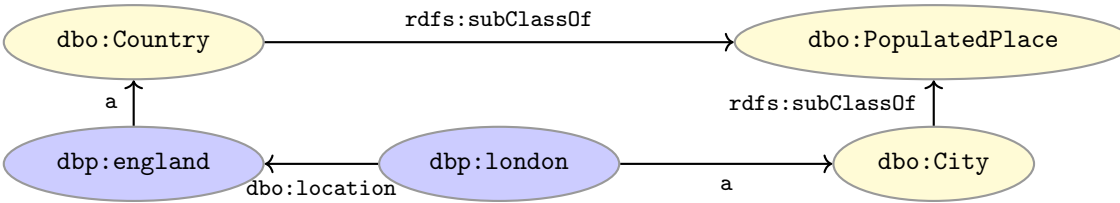
RDFS semantics

- RDFS adds the concept of **classes** which are **sets of resources**.
- RDFS has **formal semantics**.
- Entailment is a **mathematically** defined relationship between RDF(S) graphs
- Answers to **SPARQL queries** are well-defined.
- RDFS brings three types of **inference rules**:
  - type propagation
  - property propagation, and
  - domain and range reasoning.

RDFS Semantics and SPARQL

Return all Populated Places (for $\mathcal{G}$ below):

```
SELECT DISTINCT ?place WHERE {
    ?place rdf:type dbo:PopulatedPlace .
}
```

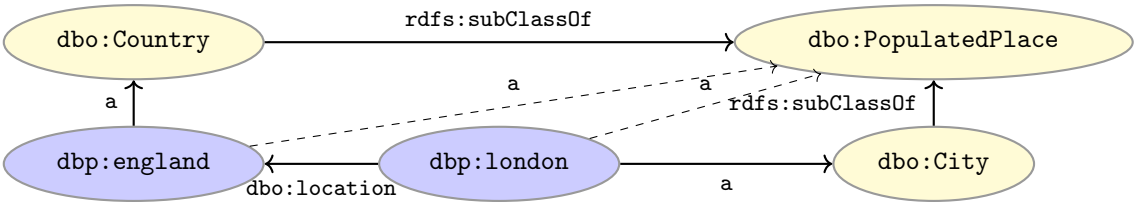


RDFS Semantics and SPARQL

Return all Populated Places (for $\mathcal{G}$ below): Query Result= {dbp:england, dbp:london}

```
SELECT DISTINCT ?place WHERE {
    ?place rdf:type dbo:PopulatedPlace .
}
```

- $\mathcal{G} \vdash$ <dbp:england a dbo:PopulatedPlace> (also entails $\models$) via rdfs9
- $\mathcal{G} \vdash$ <dbp:london a dbo:PopulatedPlace> (also entails $\models$) via rdfs9



RDFS limitations

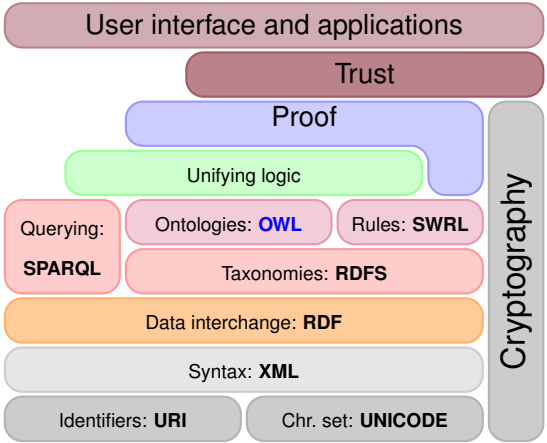
- Efficient inferencing, but **poor expressivity**.
- RDFS does **not** allow for **complex definitions**, other than multiple inheritance. We cannot express...
  - **negation** in RDFS,
  - **local scope** of properties,
  - **property characteristics** (inverse, transitivity, symmetry, etc.),
  - **complex concepts** (built from other entities).

RDFS limitations

- Efficient inferencing, but **poor expressivity**.
- RDFS does **not** allow for **complex definitions**, other than multiple inheritance. We cannot express...
  - **negation** in RDFS,
  - **local scope** of properties,
  - **property characteristics** (inverse, transitivity, symmetry, etc.),
  - **complex concepts** (built from other entities).
- **No clear boundaries** between ...
  - ontology/schema and data.
  - “object” and “data properties”.

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**
  - A knowledge representation language for the Web.
  - **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).

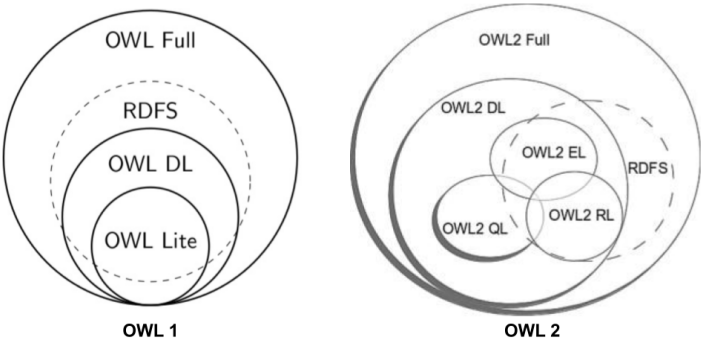
Description Logics and OWL

- **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

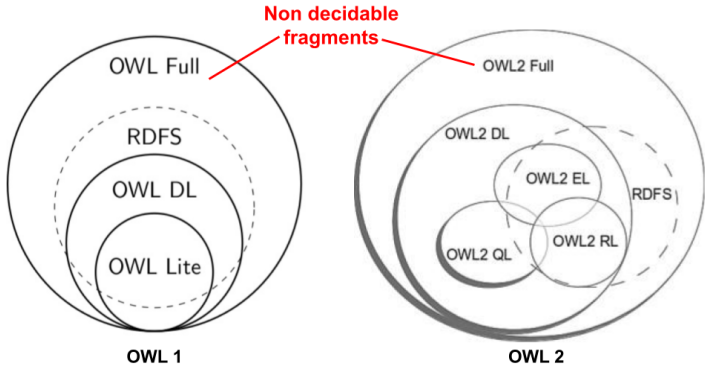
- **Origin:** semantic networks and other graph-based models and the attempt to formalise them with First Order Logic (FOL).
- Core reasoning problems in **FOL** are **undecidable**.
- **Trade-off** between **expressiveness** and computational properties
- **Description Logics** (DL):
  - Family of knowledge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



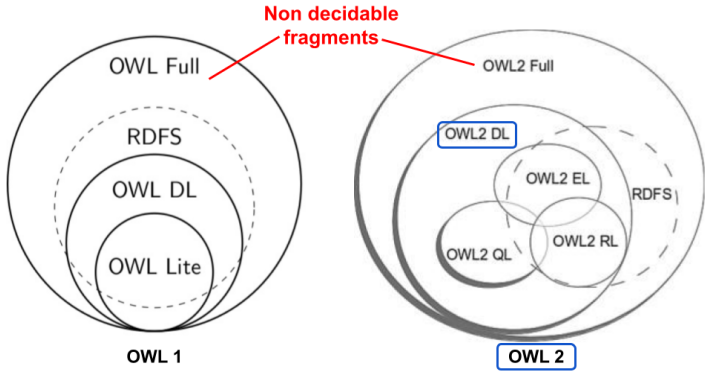
Olivier Cure and Guillaume Blin. RDF Database Systems (Chapter 3). 2015. Elsevier.

OWL 1, OWL 2 (profiles) and RDFS



(*) **Reasoning in OWL 2** will partially get the same consequences as in the RDFS inference rules and many more.

OWL 1, OWL 2 (profiles) and RDFS



(*) **Reasoning in OWL 2** will partially get the same consequences as in the RDFS inference rules and many more.

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.

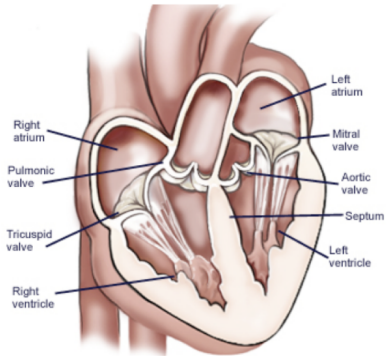
Ontologies as domain models (i)

- 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.

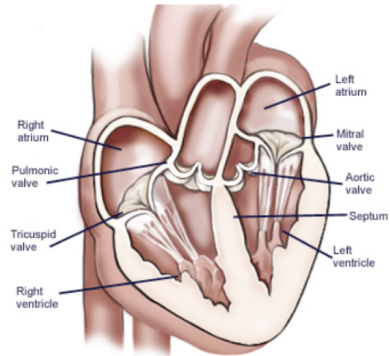
Ontologies as domain models (ii)

- include **vocabulary** relevant to a domain (*e.g.*, with RDF)
- specify meaning (**semantics**) of terms (*e.g.*, with RDFS)
  - 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)
- specify meaning (**semantics**) of terms (*e.g.*, with RDFS)
  - 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)



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.
- For example:
  - $\texttt{dbo:City}^{\mathcal{I}} = \{\texttt{dbp:london}^{\mathcal{I}}\}$
  - $\texttt{dbo:Country}^{\mathcal{I}} = \{\texttt{dbp:england}^{\mathcal{I}}\}$
  - $\texttt{dbo:PopulatedPlace}^{\mathcal{I}} = \{\texttt{dbp:london}^{\mathcal{I}}, \texttt{dbp:england}^{\mathcal{I}}\}$
  - $\texttt{dbo:location}^{\mathcal{I}} = \{\langle \texttt{dbp:london}^{\mathcal{I}}, \texttt{dbp:england}^{\mathcal{I}} \rangle\}$

Description Logics Syntax (first steps)

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`: used instead of `rdfs: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`: used instead of `rdfs: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`, set of concrete individuals (instead of `rdfs:Resource`).

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

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 $\emptyset$ (set of all individuals)
- For every `owl:Class` C, `owl:Nothing` is a subclass of C

OWL 2 entities: properties

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

- `owl:DatatypeProperty`. Target 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`. Target 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 (big difference wrt RDFS).
  - **Negation**: $\neg E$
  - **Intersection**: $E \sqcap F$
  - **Union**: $E \sqcup F$
  - **Property restrictions**. (*e.g.*, local scope for domain and ranges: $\exists R.E$, $\forall R.E$)

OWL 2 TBox Axioms

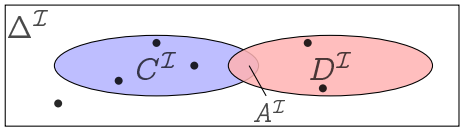
- The TBox (excluding the RBox) is composed by:
  - **Subsumption axioms:** $C \sqsubseteq D$ ($C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$)
  - **Equivalence axioms:** $C \equiv D$ ($C^{\mathcal{I}} = D^{\mathcal{I}}$)

- C and D can be **named concepts** (*e.g.*, `dbo:City`) or **complex concepts** built from others (big difference wrt RDFS).
  - **Negation**: $\neg E$
  - **Intersection**: $E \sqcap F$
  - **Union**: $E \sqcup F$
  - **Property restrictions**. (*e.g.*, local scope for domain and ranges: $\exists R.E$, $\forall R.E$)

- We will focus on the cases where the **LHS concept is atomic**.

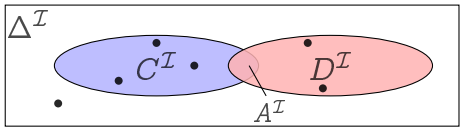
Union and Intersection

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

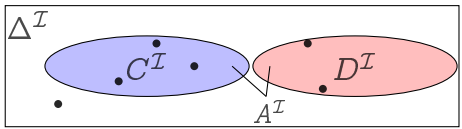


Union and Intersection

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

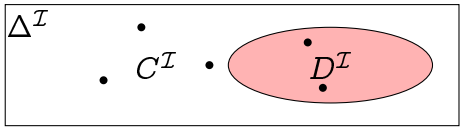


- $A \sqsubseteq C \sqcup D$.
  - *e.g.*, $\texttt{Wine} \sqsubseteq \texttt{RedWine} \sqcup \texttt{WhiteWine}$.



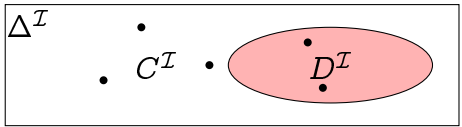
Negation and Disjointness

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

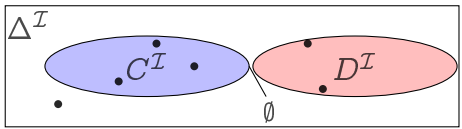


Negation and Disjointness

- $C \sqsubseteq \neg D$: "A C is not a D."
  - *e.g.*, `VegetarianTopping` $\sqsubseteq \neg$`MeatTopping`.



- $C \sqcap D \sqsubseteq \bot$: "Nothing is both a C and a D."
  - Equivalent to $C \sqsubseteq \neg D$ (and $D \sqsubseteq \neg C$).

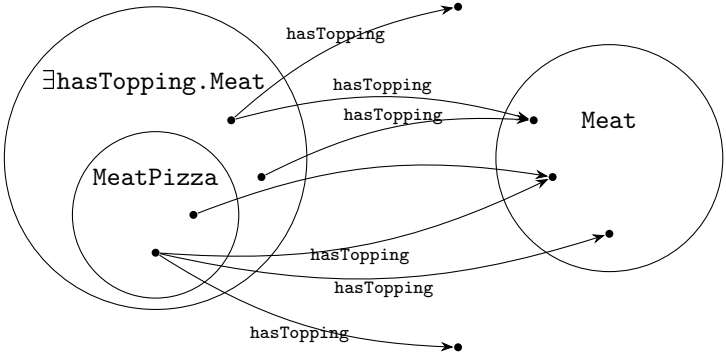


Existential Restrictions

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



Existential Restrictions and `rdfs:domain`

Domain: (global scope for R)

- If R has the *domain* C: (`R rdfs:domain C`)
- then anything from which one can go by R is in C.
- Domain can also be expressed as: $\exists R.\top \sqsubseteq C$
- $\exists$`hasTopping`.$\top \sqsubseteq$ `Pizza`

Existential Restrictions and `rdfs:domain`

Domain: (global scope for R)

- If R has the *domain* C: (`R rdfs:domain C`)
- then anything from which one can go by R is in C.
- Domain can also be expressed as: $\exists R.\top \sqsubseteq C$
- $\exists$`hasTopping`.$\top \sqsubseteq$ `Pizza`

Local scope for R:

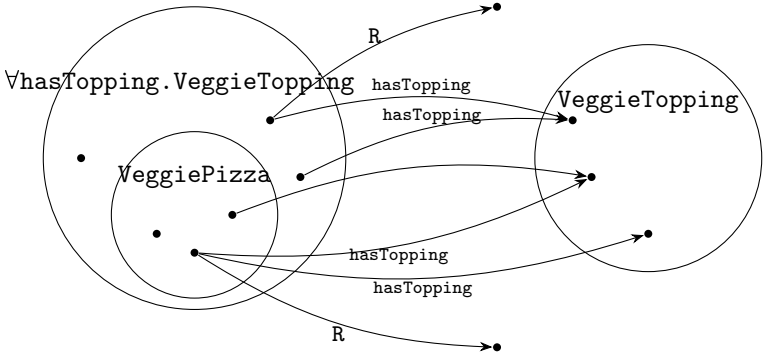
- `Pizza` $\sqsubseteq \exists$`hasTopping.Topping`

Universal Restrictions

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

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.*, `VeggiePizza` $\sqsubseteq \forall$`hasTopping.VeggieTopping`



Universal Restrictions and `rdfs:range`

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, or
- Range can also be expressed as $\top \sqsubseteq \forall R.C$.
- $\top \sqsubseteq \forall \texttt{hasTopping}.\texttt{VeggieTopping}$.

Local scope for R:

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

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

Modeling with OWL 2: RBox

Property axioms

- subsumption:

  $\texttt{hasBrother} \sqsubseteq \texttt{hasSibling}$
- equivalence:

  $\texttt{hasLocation} \equiv \texttt{locatedIn}$
- disjointness:

  $\texttt{hasLocation} \sqcap \texttt{hasBrother} \sqsubseteq \bot_{\text{role}}$
- 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`

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`
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`
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:
    `Person` $\sqcap$ ($\exists$`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)`
  Triple: `:ernesto :teaches :inm713`

ABox: Assertional axioms

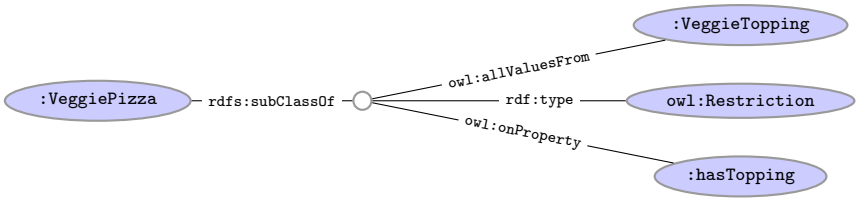
Contains:

- 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)`
  Triple: `:ernesto :teaches :inm713`
- Equality and non-equality between individuals (new from RDF(S)).
  - DL: `ernesto` $=$ `ejr`, `ernesto` $\neq$ `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:

```
: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

Next Session

Next Session on OWL

- SPARQL 1.1
- OWL 2 profiles
- Reasoning and Entailment Regimes

Laboratory: Modelling OWL 2 Ontologies with Protégé

Laboratory with OWL

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

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`