



---

# OWL 2 Profiles, SPARQL 1.1 and Entailment Regimes

**Ernesto Jiménez-Ruiz** Lecturer in Artificial Intelligence

---

---

# Before we start...

# Where are we?

- ✓ Introduction.
- ✓ RDF-based knowledge graphs.
- ✓ SPARQL 1.0
- ✓ RDFS Semantics and RDF(S)-based knowledge graphs.
- ✓ OWL (2) ontology language. Focus on modelling.
- ✓ Application to Data Science.

# Where are we?

- ✓ Introduction.
  - ✓ RDF-based knowledge graphs.
  - ✓ SPARQL 1.0
  - ✓ RDFS Semantics and RDF(S)-based knowledge graphs.
  - ✓ OWL (2) ontology language. Focus on modelling.
  - ✓ Application to Data Science.
7. **OWL 2 Profiles, SPARQL 1.1 and Entailment Regimes** (today).

# Where are we?

- ✓ Introduction.
  - ✓ RDF-based knowledge graphs.
  - ✓ SPARQL 1.0
  - ✓ RDFS Semantics and RDF(S)-based knowledge graphs.
  - ✓ OWL (2) ontology language. Focus on modelling.
  - ✓ Application to Data Science.
7. **OWL 2 Profiles, SPARQL 1.1 and Entailment Regimes** (today).
  8. Ontology Alignment (March 17)
  9. Machine Learning and Knowledge Graphs (March 24).
  10. Graph Database Solutions and [Invited Talks](#) (March 31).

---

# OWL 2 Reasoning and Profiles

# Recap: OWL (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/>



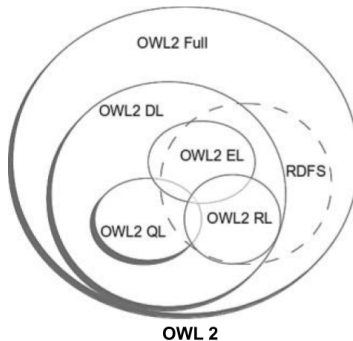
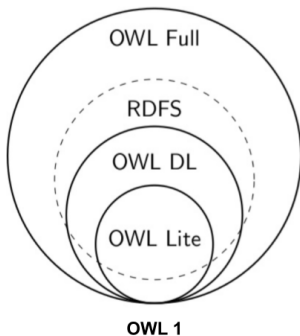
# Recap: OWL (The Web Ontology Language)

- A **W3C recommendation**:
  - OWL 1 (2004): <http://www.w3.org/TR/owl-ref/>
  - OWL 2 (2009): <https://www.w3.org/TR/owl2-overview/>
- OWL semantics based on **Description Logics (DL)**.
  - Family of knowledge representation languages
  - Decidable subset of First Order logic (FOL)
  - Original called: **Terminological language** or **concept language**



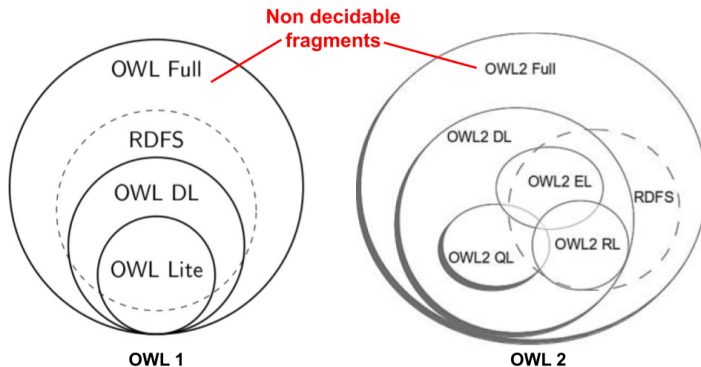


# Recap: OWL 1, OWL 2 (profiles) and RDFS



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

# Recap: 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.

## Recap: Automated Reasoning

- 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?`).

## Recap: Automated Reasoning

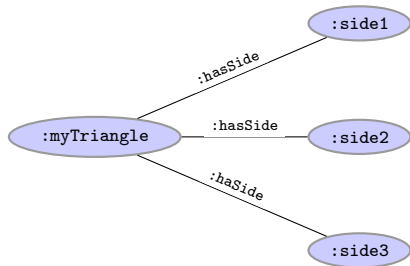
- Formal semantics allows the automatic deduction of **new facts**.
- Also allows us to perform checks that aim to detect the **correctness** of the designed model (*e.g.*, `:dolphin` is a `:Fish`?).
- Possibly in the form of **obvious errors**:
  - `:Mammal` and `:Fish` are disjoint classes.
  - `:dolphin`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

## Recap: Automated Reasoning

- 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.

## Recap: OWL 2 and Open World Assumption

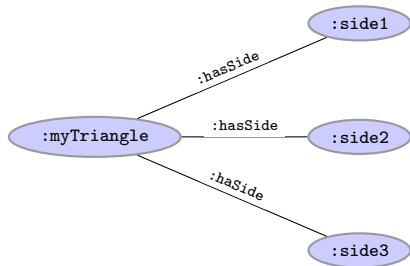
- `:Triangle EquivalentTo :hasSide exactly 3 :Side`



- is `:myTriangle` a `:Triangle`?

## Recap: OWL 2 and Open World Assumption

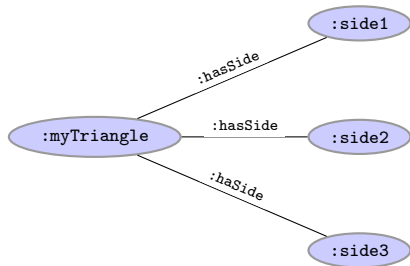
- `:Triangle EquivalentTo :hasSide exactly 3 :Side`



- is `:myTriangle` a `:Triangle`?

## Recap: OWL 2 and Open World Assumption

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



- is `:myTriangle` a `:Triangle`? **No, because of OWA and NUNA.**
- **Solution:** reasoning in OWL can be complemented with SPARQL queries (in this case with aggregates) → SPARQL 1.1



## Recap: OWL 2 Axioms into Boxes

- Traditionally OWL 2 axioms are put in boxes.
- The **TBox** (terminological knowledge)
  - Typically independent of any actual instance data.
  - Property axioms are also referred to as **RBox**
- The **ABox** (assertional knowledge)
  - Contains facts about concrete instance.

# OWL 2 TBox Reasoning

(Standard) Reasoning tasks that use only the TBox  $\mathcal{T}^\dagger$

- Concept **unsatisfiability**: Given  $C$ , does  $\mathcal{T} \models C \sqsubseteq \perp$ ? (i.e.,  $\mathcal{C}^\mathcal{I} = \emptyset$ )

† (**Model-Theoretic Semantics**) The answer to ‘does  $\mathcal{T} \models \alpha$ ?’ will be positive if for each interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models \mathcal{T}$ ,  $\mathcal{I} \models \alpha$  too.

# OWL 2 TBox Reasoning

(Standard) Reasoning tasks that use only the TBox  $\mathcal{T}^\dagger$

- Concept **unsatisfiability**: Given  $C$ , does  $\mathcal{T} \models C \sqsubseteq \perp$ ? (i.e.,  $\mathcal{C}^\mathcal{I} = \emptyset$ )
- Concept **subsumption**: Given  $C$  and  $D$ , does  $\mathcal{T} \models C \sqsubseteq D$ ? (i.e.,  $\mathcal{C}^\mathcal{I} \subseteq \mathcal{D}^\mathcal{I}$ )

† (**Model-Theoretic Semantics**) The answer to ‘does  $\mathcal{T} \models \alpha$ ?’ will be positive if for each interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models \mathcal{T}$ ,  $\mathcal{I} \models \alpha$  too.

# OWL 2 TBox Reasoning

(Standard) Reasoning tasks that use only the TBox  $\mathcal{T}^\dagger$

- Concept **unsatisfiability**: Given  $C$ , does  $\mathcal{T} \models C \sqsubseteq \perp$ ? (i.e.,  $\mathcal{C}^\mathcal{I} = \emptyset$ )
- Concept **subsumption**: Given  $C$  and  $D$ , does  $\mathcal{T} \models C \sqsubseteq D$ ? (i.e.,  $\mathcal{C}^\mathcal{I} \subseteq \mathcal{D}^\mathcal{I}$ )
- Concept **equivalence**: Given  $C$  and  $D$ , does  $\mathcal{T} \models C \equiv D$ ? (i.e.,  $\mathcal{C}^\mathcal{I} = \mathcal{D}^\mathcal{I}$ )

† **(Model-Theoretic Semantics)** The answer to ‘does  $\mathcal{T} \models \alpha$ ?’ will be positive if for each interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models \mathcal{T}$ ,  $\mathcal{I} \models \alpha$  too.

# OWL 2 TBox Reasoning

(Standard) Reasoning tasks that use only the TBox  $\mathcal{T}^\dagger$

- Concept **unsatisfiability**: Given  $C$ , does  $\mathcal{T} \models C \sqsubseteq \perp$ ? (i.e.,  $\mathcal{C}^\mathcal{I} = \emptyset$ )
- Concept **subsumption**: Given  $C$  and  $D$ , does  $\mathcal{T} \models C \sqsubseteq D$ ? (i.e.,  $\mathcal{C}^\mathcal{I} \subseteq \mathcal{D}^\mathcal{I}$ )
- Concept **equivalence**: Given  $C$  and  $D$ , does  $\mathcal{T} \models C \equiv D$ ? (i.e.,  $\mathcal{C}^\mathcal{I} = \mathcal{D}^\mathcal{I}$ )
- Concept **disjointness**: Given  $C$  and  $D$ , does  $\mathcal{T} \models C \sqcap D \sqsubseteq \perp$ ? (i.e.,  $\mathcal{C}^\mathcal{I} \cap \mathcal{D}^\mathcal{I} \subseteq \emptyset$ )

† (**Model-Theoretic Semantics**) The answer to ‘does  $\mathcal{T} \models \alpha$ ?’ will be positive if for each interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models \mathcal{T}$ ,  $\mathcal{I} \models \alpha$  too.

# OWL 2 ABox Reasoning

(Standard) Reasoning tasks that involve both the TBox  $\mathcal{T}$  and Abox  $\mathcal{A}$

- **Consistency:**

Is there a model for  $(\mathcal{T}, \mathcal{A})$ ? i.e., is there an interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ ?

† **(Model-Theoretic Semantics)** The answer to ‘does  $(\mathcal{T}, \mathcal{A}) \models \alpha$ ?’ will be positive if for each interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ ,  $\mathcal{I} \models \alpha$  too.

# OWL 2 ABox Reasoning

(Standard) Reasoning tasks that involve both the TBox  $\mathcal{T}$  and Abox  $\mathcal{A}$

- **Consistency:**

Is there a model for  $(\mathcal{T}, \mathcal{A})$ ? i.e., is there an interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ ?

- Concept **membership**: Given  $C$  and  $a$ , does  $(\mathcal{T}, \mathcal{A}) \models C(a)$ ?<sup>†</sup>

<sup>†</sup> **(Model-Theoretic Semantics)** The answer to ‘does  $(\mathcal{T}, \mathcal{A}) \models \alpha$ ?’ will be positive if for each interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ ,  $\mathcal{I} \models \alpha$  too.

# OWL 2 ABox Reasoning

(Standard) Reasoning tasks that involve both the TBox  $\mathcal{T}$  and Abox  $\mathcal{A}$

- **Consistency:**

Is there a model for  $(\mathcal{T}, \mathcal{A})$ ? i.e., is there an interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ ?

- Concept **membership**: Given  $C$  and  $a$ , does  $(\mathcal{T}, \mathcal{A}) \models C(a)$ ?<sup>†</sup>

- **Role assertion**: Given  $R$ ,  $a$  and  $b$ , does  $(\mathcal{T}, \mathcal{A}) \models R(a, b)$ ?<sup>†</sup>

<sup>†</sup> **(Model-Theoretic Semantics)** The answer to ‘does  $(\mathcal{T}, \mathcal{A}) \models \alpha$ ?’ will be positive if for each interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ ,  $\mathcal{I} \models \alpha$  too.



## OWL 2 ABox Reasoning

(Standard) Reasoning tasks that involve both the TBox  $\mathcal{T}$  and Abox  $\mathcal{A}$

- **Consistency:**

Is there a model for  $(\mathcal{T}, \mathcal{A})$ ? i.e., is there an interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ ?

- Concept **membership**: Given  $C$  and  $a$ , does  $(\mathcal{T}, \mathcal{A}) \models C(a)$ ?<sup>†</sup>

- **Role assertion**: Given  $R$ ,  $a$  and  $b$ , does  $(\mathcal{T}, \mathcal{A}) \models R(a, b)$ ?<sup>†</sup>

- **Retrieval**: Given  $C$ , find all  $a$  such that  $(\mathcal{T}, \mathcal{A}) \models C(a)$ .

<sup>†</sup> **(Model-Theoretic Semantics)** The answer to ‘does  $(\mathcal{T}, \mathcal{A}) \models \alpha$ ?’ will be positive if for each interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ ,  $\mathcal{I} \models \alpha$  too.

# OWL 2 ABox Reasoning

(Standard) Reasoning tasks that involve both the TBox  $\mathcal{T}$  and Abox  $\mathcal{A}$

- **Consistency:**

Is there a model for  $(\mathcal{T}, \mathcal{A})$ ? i.e., is there an interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ ?

- Concept **membership**: Given  $C$  and  $a$ , does  $(\mathcal{T}, \mathcal{A}) \models C(a)$ ?<sup>†</sup>

- **Role assertion**: Given  $R$ ,  $a$  and  $b$ , does  $(\mathcal{T}, \mathcal{A}) \models R(a, b)$ ?<sup>†</sup>

- **Retrieval**: Given  $C$ , find all  $a$  such that  $(\mathcal{T}, \mathcal{A}) \models C(a)$ .

- Conjunctive Query Answering (**SPARQL**).

<sup>†</sup> (**Model-Theoretic Semantics**) The answer to ‘does  $(\mathcal{T}, \mathcal{A}) \models \alpha$ ?’ will be positive if for each interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ ,  $\mathcal{I} \models \alpha$  too.

## OWL 2 Reasoning Algorithms

- Reasoning in OWL 2 is typically based on **(Hyper)Tableau Reasoning Algorithms** (tableau = truth tree)
- Reasoning tasks **reduced to (un)satisfiability**.
- Algorithm tries to construct an **abstraction** of a model.

Chapter 5: Foundations of Semantic Web Technologies. CRC Press 2009

Seminars by Prof. Ian Horrocks: <http://www.cs.ox.ac.uk/people/ian.horrocks/Seminars/seminars.html>

## OWL 2 Reasoning Algorithms

- Reasoning in OWL 2 is typically based on **(Hyper)Tableau Reasoning Algorithms** (tableau = truth tree)
- Reasoning tasks **reduced to (un)satisfiability**.
- Algorithm tries to construct an **abstraction** of a model.
- State-of-the-art algorithms:
  - *e.g.*, **HermiT** (default option in Protégé).
  - Implement a number of (search) **optimisations**.
  - **Effective** with many realistic ontologies

Chapter 5: Foundations of Semantic Web Technologies. CRC Press 2009

Seminars by Prof. Ian Horrocks: <http://www.cs.ox.ac.uk/people/ian.horrocks/Seminars/seminars.html>

# Tractability Problems with OWL 2 Reasoning

- Problems with **very large** and/or **cyclical ontologies**.
  - Ontologies may define hundred of thousands of terms (*e.g.*, SNOMED CT)
  - Large number of tests for classification (each test can lead to the construction of very large models).

Computational properties: [https://www.w3.org/TR/owl2-profiles/#Computational\\_Properties](https://www.w3.org/TR/owl2-profiles/#Computational_Properties)

Seminars by Prof. Ian Horrocks: <http://www.cs.ox.ac.uk/people/ian.horrocks/Seminars/seminars.html>

# Tractability Problems with OWL 2 Reasoning

- Problems with **very large** and/or **cyclical ontologies**.
  - Ontologies may define hundred of thousands of terms (*e.g.*, SNOMED CT)
  - Large number of tests for classification (each test can lead to the construction of very large models).
- Problems with **medium/large** data sets (ABoxes)
  - OWL 2 Reasoners typically optimized for TBox reasoning tasks.
  - Data also brings additional complexity.

Computational properties: [https://www.w3.org/TR/owl2-profiles/#Computational\\_Properties](https://www.w3.org/TR/owl2-profiles/#Computational_Properties)

Seminars by Prof. Ian Horrocks: <http://www.cs.ox.ac.uk/people/ian.horrocks/Seminars/seminars.html>

---

# OWL 2 profiles

## OWL 2 profiles

- OWL 2 has various **profiles** that correspond to different DLs.
- These profiles have very interesting **computational properties**.

OWL 2 Validator: <http://visualdataweb.de/validator/>



## OWL 2 profiles

- OWL 2 has various **profiles** that correspond to different DLs.
- These profiles have very interesting **computational properties**.
  - **OWL 2 QL**:
    - Specifically designed for efficient database integration.

OWL 2 Validator: <http://visualdataweb.de/validator/>

## OWL 2 profiles

- OWL 2 has various **profiles** that correspond to different DLs.
- These profiles have very interesting **computational properties**.
  - **OWL 2 QL**:
    - Specifically designed for efficient database integration.
  - **OWL 2 EL**:
    - A lightweight language with polynomial time reasoning.

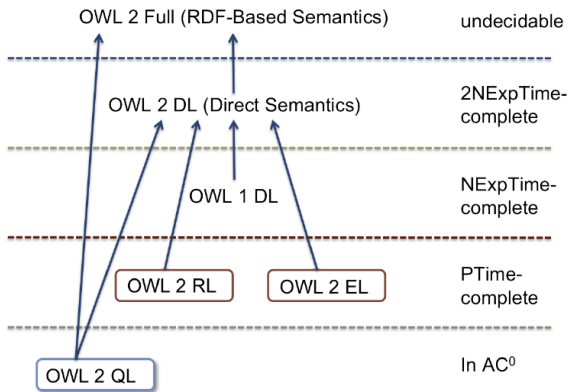
OWL 2 Validator: <http://visualdataweb.de/validator/>

## OWL 2 profiles

- OWL 2 has various **profiles** that correspond to different DLs.
- These profiles have very interesting **computational properties**.
  - **OWL 2 QL**:
    - Specifically designed for efficient database integration.
  - **OWL 2 EL**:
    - A lightweight language with polynomial time reasoning.
  - **OWL 2 RL**:
    - Designed for compatibility with rule-based inference tools.
    - Efficient reasoning with large datasets.

OWL 2 Validator: <http://visualdataweb.de/validator/>

# Data Complexity OWL 2 Profiles



<https://www.w3.org/TR/owl2-profiles/>

# OWL EL profile (i)

Based on the DL  $\mathcal{EL}^{++}$ . Concept descriptions, simplified

$C, D \rightarrow$	$A$		(atomic concept)
	$\top$		(universal concept)
	$\perp$		(bottom concept)
	$\{a\}$		( <i>singular</i> enumeration)
	$C \sqcap D$		(intersection)
	$\exists R.C$		(existential restriction)

## Axioms

- $C \sqsubseteq D$  and  $C \equiv D$  for concept descriptions  $D$  and  $C$ .
- $P \sqsubseteq Q$  and  $P \equiv Q$  for roles  $P, Q$ . Also Domain and Range.
- $C(a)$  and  $R(a, b)$  for concept  $C$ , role  $R$  and individuals  $a, b$ .

## OWL EL Profile (ii)

- ✓ Standard reasoning tasks in  $P$  time
- ✓ Very good for large ontologies.
- ✓ Used in many biomedical ontologies (*e.g.*, SNOMED CT).

Not supported features, simplified:

- ✗ negation (but  $C \sqcap D \sqsubseteq \perp$  possible)
- ✗ disjunction
- ✗ universal quantification and cardinalities
- ✗ inverse roles and some role characteristics
- ✗ reduced list of datatypes

## OWL EL Profile (iii)

- Reasoning can be performed via saturation<sup>†</sup> (*i.e.*, inference rules).
- For example:

$$\frac{A \sqsubseteq B \quad B \sqsubseteq C}{A \sqsubseteq C}$$
$$\frac{A \sqsubseteq \exists R.B \quad \exists S.B \sqsubseteq C \quad S \sqsubseteq R}{A \sqsubseteq C}$$

<sup>†</sup> Using a saturation-based approach over an OWL 2 ontology is not possible.

ELK reasoner (also available as Protégé plugin): <https://github.com/liveontologies/elk-reasoner/wiki>

# OWL QL Profile (i)

## Based on DL-Lite<sub>R</sub>. Concept descriptions, simplified

$C, C' \rightarrow$	$A$		(atomic concept)
	$\exists R. \top$		(existential restriction with $\top$ only)
$D, D' \rightarrow$	$A$		(atomic concept)
	$\exists R. D$		(existential restriction)
	$\neg D$		(negation)
	$D \sqcap D'$		(intersection)

## Axioms

- $C \sqsubseteq D$  for concept descriptions  $D$  and  $C$  (and  $C \equiv C'$ ).
- $P \sqsubseteq Q$  and  $P \equiv Q$  for roles  $P, Q$ . Also Domain and Range.
- $C(a)$  and  $R(a, b)$  for concept  $C$ , role  $R$  and individuals  $a, b$ .



## OWL QL Profile (ii)

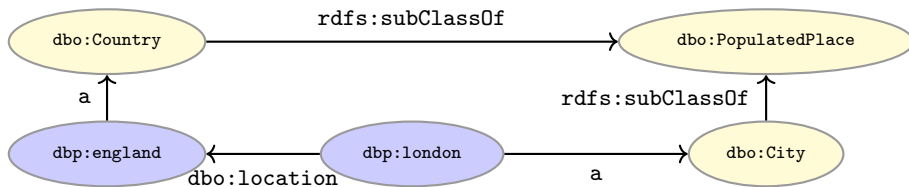
- ✓ Required language so that queries can be rewritten and then translated to SQL.
- ✓ Used in Ontology Based Data Access (OBDA).

Not supported, simplified:

- ✗ disjunction
- ✗ universal quantification, cardinalities, and functional roles
- ✗ `= (SameIndividual)`
- ✗ enumerations (closed classes)
- ✗ subproperties of chains, transitivity
- ✗ reduced list of datatypes

## OWL QL Profile (iii)

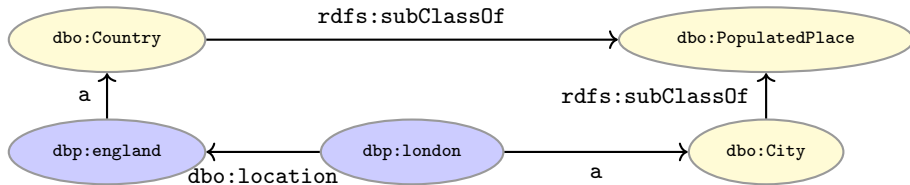
- Reasoning is performed via backward chaining (e.g., rewriting of a given query  $Q$  into  $Q'$  via the ontology axioms). For example:



## OWL QL Profile (iii)

- Reasoning is performed via backward chaining (e.g., rewriting of a given query  $Q$  into  $Q'$  via the ontology axioms). For example:

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

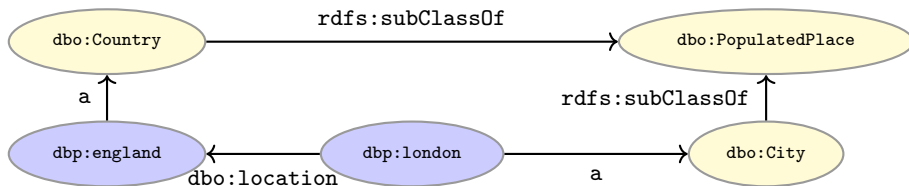


## OWL QL Profile (iii)

- Reasoning is performed via backward chaining (e.g., rewriting of a given query  $Q$  into  $Q'$  via the ontology axioms). For example:

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

$Q'$ : `SELECT DISTINCT ?place WHERE {  
 {?place rdf:type dbo:PopulatedPlace .}  
 UNION {?place rdf:type dbo:Country .}  
 UNION {?place rdf:type dbo:City .} }`



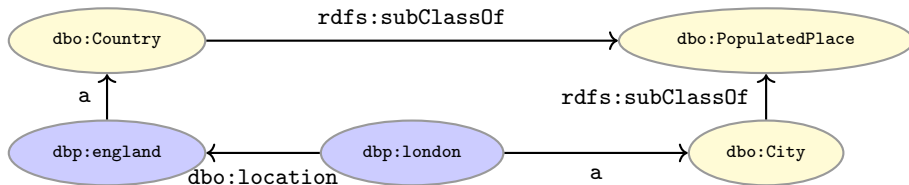
## OWL QL Profile (iii)

- Reasoning is performed via backward chaining (e.g., rewriting of a given query  $Q$  into  $Q'$  via the ontology axioms). For example:

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

$Q'$ : `SELECT DISTINCT ?place WHERE {  
 {?place rdf:type dbo:PopulatedPlace .}  
 UNION {?place rdf:type dbo:Country .}  
 UNION {?place rdf:type dbo:City .} }`

$Q'$  Result= {dbp:england, dbp:london}



# OWL 2 RL Profile (i)

Based on Description Logic Programs (DLP). Concept descriptions:

$C, C' \rightarrow$	$A$		(atomic concept)
	$C \sqcap C'$		(intersection)
	$C \sqcup C'$		(union)
	$\exists R.C$		(existential restriction)
$D, D' \rightarrow$	$A$		(atomic concept)
	$D \sqcap D'$		(intersection)
	$\forall R.D$		(universal restriction)

## Axioms

- $C \sqsubseteq D$ ,  $C \equiv C'$ ,  $\top \sqsubseteq \forall R.D$ ,  $\top \sqsubseteq \forall R^-.D$ ,  $R \sqsubseteq P$ ,  $R \equiv P^-$  and  $R \equiv P$  for roles  $R, P$  and concept descriptions  $C, C'$  and  $D$ . Also Domain and Range.
- $C(a)$  and  $R(a, b)$  for concept  $C$ , role  $R$  and individuals  $a, b$ .

## OWL 2 RL Profile (ii)

- ✗ Puts syntactic constraints in the way in which constructs are used (i.e., syntactic subset of OWL 2).
- ✗ Imposes a reduced list of allowed datatypes
- ✓ OWL 2 RL axioms can be directly translated into datalog rules
- ✓ Enables desirable computational properties using rule-based reasoning engines.

## OWL 2 RL Profile (iii)

- Reasoning via full materialisation of the graph, similarly to RDFS inference rules. *e.g.*,:

$$\frac{p1 \text{ owl:inverseOf } p2 . \quad ?x \text{ ?p1 } ?y .}{?y \text{ ?p2 } ?x .}$$

W3C: [https://www.w3.org/TR/owl2-profiles/#Reasoning\\_in\\_OWL\\_2\\_RL\\_and\\_RDF\\_Graphs\\_using\\_Rules](https://www.w3.org/TR/owl2-profiles/#Reasoning_in_OWL_2_RL_and_RDF_Graphs_using_Rules)

GraphDB: <https://graphdb.ontotext.com/documentation/standard/reasoning.html>

RDFFox: A Highly-Scalable RDF Store. ISWC 2015. <https://www.oxfordsemantic.tech/product>



---

# SPARQL 1.1

# SPARQL

- SPARQL Protocol And RDF Query Language
- Standard language to query graph data represented as **RDF triples**
- W3C Recommendations
  - **SPARQL 1.0**: W3C Recommendation 15 January 2008
  - **SPARQL 1.1**: W3C Recommendation 21 March 2013

# SPARQL

- SPARQL Protocol And RDF Query Language
- Standard language to query graph data represented as **RDF triples**
- W3C Recommendations
  - **SPARQL 1.0**: W3C Recommendation 15 January 2008
  - **SPARQL 1.1**: W3C Recommendation 21 March 2013
- In this lecture we will learn about the extensions in SPARQL 1.1.
- Documentation:
  - Syntax and semantics of the SPARQL query language for RDF.  
<https://www.w3.org/TR/sparql11-overview/>

## Recap: Components of a SPARQL (1.0) query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?costar
FROM <http://dbpedia.org>
WHERE {
    ?jd foaf:name "Johnny Depp"@en .
    ?m dbo:starring ?jd .
    ?m dbo:starring ?other .
    ?other foaf:name ?costar .
    FILTER (STR(?costar)!="Johnny Depp")
}
ORDER BY ?costar
```

# Recap: Components of a SPARQL (1.0) query

Prologue: prefix definitions

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX dbo: <http://dbpedia.org/ontology/>
```

```
SELECT DISTINCT ?costar
```

```
FROM <http://dbpedia.org>
```

```
WHERE {
```

```
    ?jd foaf:name "Johnny Depp"@en .
```

```
    ?m dbo:starring ?jd .
```

```
    ?m dbo:starring ?other .
```

```
    ?other foaf:name ?costar .
```

```
    FILTER (STR(?costar)!="Johnny Depp")
```

```
}
```

```
ORDER BY ?costar
```

## Recap: Components of a SPARQL (1.0) query

**Results:** (1) variable list, (2) query type (SELECT, ASK, CONSTRUCT, DESCRIBE), (3) remove duplicates (DISTINCT, REDUCED)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX dbo: <http://dbpedia.org/ontology/>
```

```
SELECT DISTINCT ?costar
```

```
FROM <http://dbpedia.org>
```

```
WHERE {
```

```
    ?jd foaf:name "Johnny Depp"@en .
```

```
    ?m dbo:starring ?jd .
```

```
    ?m dbo:starring ?other .
```

```
    ?other foaf:name ?costar .
```

```
    FILTER (STR(?costar)!="Johnny Depp")
```

```
}
```

```
ORDER BY ?costar
```

# Recap: Components of a SPARQL (1.0) query

## Dataset specification

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dbo:  <http://dbpedia.org/ontology/>
SELECT DISTINCT ?costar
FROM <http://dbpedia.org>
WHERE {
    ?jd foaf:name "Johnny Depp"@en .
    ?m dbo:starring ?jd .
    ?m dbo:starring ?other .
    ?other foaf:name ?costar .
    FILTER (STR(?costar)!="Johnny Depp")
}
ORDER BY ?costar
```

# Recap: Components of a SPARQL (1.0) query

Query pattern: graph pattern to be matched + filters

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dbo:  <http://dbpedia.org/ontology/>
SELECT DISTINCT ?costar
FROM <http://dbpedia.org>
WHERE {
    ?jd foaf:name "Johnny Depp"@en .
    ?m dbo:starring ?jd .
    ?m dbo:starring ?other .
    ?other foaf:name ?costar .
    FILTER (STR(?costar)!="Johnny Depp")
}
ORDER BY ?costar
```



# Recap: Components of a SPARQL (1.0) query

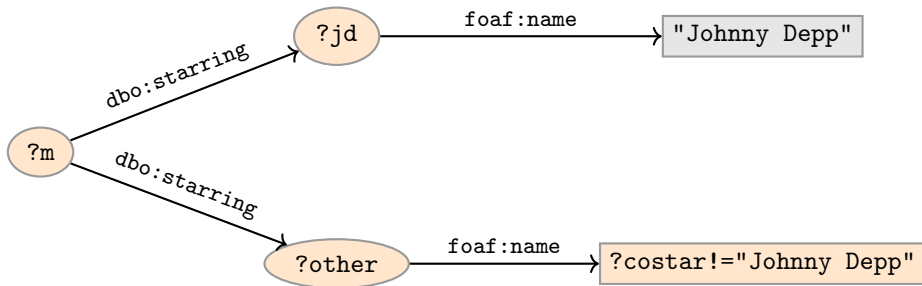
**Solution modifiers:** ORDER BY, LIMIT, OFFSET

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dbo:  <http://dbpedia.org/ontology/>
SELECT DISTINCT ?costar
FROM <http://dbpedia.org>
WHERE {
    ?jd foaf:name "Johnny Depp"@en .
    ?m dbo:starring ?jd .
    ?m dbo:starring ?other .
    ?other foaf:name ?costar .
    FILTER (STR(?costar)!="Johnny Depp")
}
```

**ORDER BY ?costar**

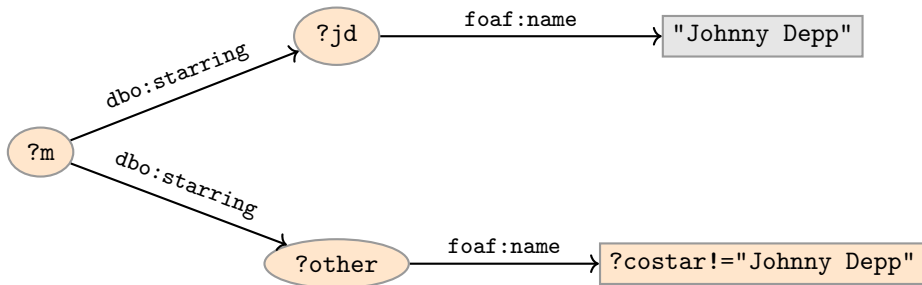
## Recap: Graph Patterns

The previous SPARQL query pattern as a graph:



## Recap: Graph Patterns

The previous SPARQL query pattern as a graph:



**Pattern matching:** assign values to variables to make this a sub-graph of the RDF graph.

## SPARQL 1.1: new features

- The new features in **SPARQL 1.1 QUERY language**:
  - Assignments and Expressions
  - Aggregates
  - Subqueries
  - Negation
  - Property paths

## SPARQL 1.1: new features

- The new features in **SPARQL 1.1 QUERY language**:
  - Assignments and Expressions
  - Aggregates
  - Subqueries
  - Negation
  - Property paths
- Specification for:
  - **SPARQL 1.1 UPDATE Language**
  - **SPARQL 1.1 Federated Queries**
  - **SPARQL 1.1 Entailment Regimes**

# Assignment and Expressions

- The value of an expression can be assigned/bound to a new variable
- Can be used in SELECT, BIND or GROUP BY clauses:  
(*expression AS ?var*)

## Expressions in SELECT clause

```
SELECT ?city (xsd:integer(?pop)/xsd:float(?area) AS ?density)
{
  ?city dbo:populationTotal ?pop .
  ?city <http://dbpedia.org/ontology/PopulatedPlace/areaTotal> ?area .
  ?city dbo:country <http://dbpedia.org/resource/United_Kingdom> .
  FILTER (xsd:float(?area)>0.0)
}
```

## Aggregates: Grouping and Filtering

- Solutions can optionally be grouped according to one or more expressions.
- To specify the group, use `GROUP BY`.
- If `GROUP BY` is not used, then only one (implicit) group

## Aggregates: Grouping and Filtering

- Solutions can optionally be grouped according to one or more expressions.
- To specify the group, use `GROUP BY`.
- If `GROUP BY` is not used, then only one (implicit) group
- To filter solutions resulting from grouping, use `HAVING`.
- `HAVING` operates over grouped solution sets, in the same way that `FILTER` operates over un-grouped ones.



# Aggregates: Example

## Actors with more than 15 movies

```
SELECT ?name (COUNT(?movie) AS ?count)
WHERE {
    ?actor foaf:name ?name .
    ?movie dbo:starring ?actor .
}
GROUP BY ?name
HAVING (COUNT(?movie) > 15)
ORDER BY DESC (?mcount)
```

# Aggregates: Example

## Actors with more than 15 movies

```
SELECT ?name (COUNT(?movie) AS ?count)
WHERE {
    ?actor foaf:name ?name .
    ?movie dbo:starring ?actor .
}
GROUP BY ?name
HAVING (COUNT(?movie) > 15)
ORDER BY DESC (?mcount)
```

† Only expressions consisting of aggregates and constants may be projected, together with variables in GROUP BY.

## Aggregates: common functions

- `Count` counts the number of times a variable has been bound.
- `Sum` sums numerical values of bound variables.
- `Avg` finds the average of numerical values of bound variables.
- `Min` finds the minimum of the numerical values of bound variables.
- `Max` finds the maximum of the numerical values of bound variables.

† Aggregates assume CWA and UNA

# Subqueries

- A way to embed SPARQL queries within other queries
- Subqueries are evaluated first and the results are projected to the outer query.

# Subqueries

- A way to embed SPARQL queries within other queries
- Subqueries are evaluated first and the results are projected to the outer query.

```
SELECT ?country ?pop (round(?pop/?worldpop*1000)/10 AS ?percentage) WHERE {  
  ?country rdf:type dbo:Country .  
  ?country dbo:populationTotal ?pop .  
  {  
    SELECT (sum(?p) AS ?worldpop) WHERE {  
      ?c rdf:type dbo:Country .  
      ?c dbo:populationTotal ?p .}  
  }  
} ORDER BY desc(?population)
```

# Negation in SPARQL 1.0

COMBINING OPTIONAL, FILTER and !BOUND:

## People without names

```
SELECT DISTINCT * WHERE {  
  ?person a foaf:Person .  
  OPTIONAL {  
    ?person foaf:name ?name .  
    FILTER (!bound(?name))  
  }  
}
```

However, this is not very easy to write.

# Negation in SPARQL 1.1: MINUS and FILTER NOT EXISTS

Two ways to do negation:

```
SELECT DISTINCT * WHERE {  
  ?person a foaf:Person .  
  MINUS { ?person foaf:name ?name }  
}
```

```
SELECT DISTINCT * WHERE {  
  ?person a foaf:Person .  
  FILTER NOT EXISTS { ?person foaf:name ?name }  
}
```

†Negation assumes CWA and UNA

## Property paths: basic motivation

- Some queries get needlessly large.
- SPARQL 1.1 define a small language to defined paths.
- Examples:
  - `uio:Ernesto foaf:knows+ ?friend` to extract all friends of friends.
  - `foaf:maker|dct:creator` instead of UNION.
  - Friend's names, `{ _:me foaf:knows/foaf:name ?friendsname }`.
  - Sum several items:  
`SELECT (sum(?cost) AS ?total) { :order :hasItem/:price ?cost }`



# Property paths: syntax

Syntax Form	Matches
<code>iri</code>	An (property) IRI. A path of length one.
<code>^elt</code>	Inverse path (object to subject).
<code>elt1 / elt2</code>	A sequence path of <code>elt1</code> followed by <code>elt2</code> .
<code>elt1   elt2</code>	A alternative path of <code>elt1</code> or <code>elt2</code> (all possibilities are tried).
<code>elt*</code>	Seq. of zero or more matches of <code>elt</code> .
<code>elt+</code>	Seq. of one or more matches of <code>elt</code> .
<code>elt?</code>	Zero or one matches of <code>elt</code> .
<code>!iri</code> or <code>!(iri<sub>1</sub>   ...   iri<sub>n</sub>)</code>	Negated property set.
<code>!^iri</code> or <code>!(^iri<sub>1</sub>   ...   ^iri<sub>n</sub>)</code>	Negation of inverse path.
<code>!(iri<sub>1</sub>   ...   iri<sub>j</sub>   ^iri<sub>j+1</sub>   ...   ^iri<sub>n</sub>)</code>	Negated combination of forward and inverse properties.
<code>(elt)</code>	A group path <code>elt</code> , brackets control precedence.

\* `elt` is a path element, which may itself be composed of path constructs (see Syntax form).

---

# SPARQL 1.1 Entailment Regimes

## OWL 2 Entailment regimes: overview

- Gives guidance for SPARQL query engines
- Basic graph pattern by means of subgraph matching: *simple entailment*
- Solutions that implicitly follow from the queried graph: *entailment regimes*
- **RDF entailment, RDF Schema entailment, D-Entailment, OWL 2 RDF-Based Semantics entailment, OWL 2 Direct Semantics entailment**, and RIF-Simple entailment
- <https://www.w3.org/TR/2013/REC-sparql11-entailment-20130321/>

# OWL 2 Entailment regimes: overview

## OWL 2 RDF-based Semantics Entailment Regime

- Direct extension of the RDFS semantics
- Interprets RDF triples directly without the need of mapping an RDF graph into OWL objects.
- Incomplete for OWL 2 and undecidable for OWL 2 Full.

## OWL 2 Direct Semantics Entailment Regime

- Decidable if some restrictions are imposed to the RDF graph and SPARQL queries.

## OWL 2 Entailment Regimes: Complexity and Profiles

- Entailment under **OWL 2 (DL) Direct Semantics** entailment is decidable, but computationally hard.

## OWL 2 Entailment Regimes: Complexity and Profiles

- Entailment under **OWL 2 (DL) Direct Semantics** entailment is decidable, but computationally hard.
- Entailment under **OWL 2 (DL) RDF-based semantics** is incomplete and undecidable for OWL 2 Full.

## OWL 2 Entailment Regimes: Complexity and Profiles

- Entailment under **OWL 2 (DL) Direct Semantics** entailment is decidable, but computationally hard.
- Entailment under **OWL 2 (DL) RDF-based semantics** is incomplete and undecidable for OWL 2 Full.
- No Direct Semantics for OWL 2 Full.

## OWL 2 Entailment Regimes: Complexity and Profiles

- Entailment under **OWL 2 (DL) Direct Semantics** entailment is decidable, but computationally hard.
- Entailment under **OWL 2 (DL) RDF-based semantics** is incomplete and undecidable for OWL 2 Full.
- No Direct Semantics for OWL 2 Full.
- **Direct Semantics for OWL 2 QL and EL** Profiles have very nice computational properties.



## OWL 2 Entailment Regimes: Complexity and Profiles

- Entailment under **OWL 2 (DL) Direct Semantics** entailment is decidable, but computationally hard.
- Entailment under **OWL 2 (DL) RDF-based semantics** is incomplete and undecidable for OWL 2 Full.
- No Direct Semantics for OWL 2 Full.
- **Direct Semantics for OWL 2 QL and EL** Profiles have very nice computational properties.
- Entailment under **OWL 2 QL and EL RDF-based semantics** is incomplete as well.

## OWL 2 Entailment Regimes: Complexity and Profiles (cont.)

- **OWL 2 RL** defines a syntactic subset of OWL 2. For RDF graphs that fall into this syntactic subset:

## OWL 2 Entailment Regimes: Complexity and Profiles (cont.)

- **OWL 2 RL** defines a syntactic subset of OWL 2. For RDF graphs that fall into this syntactic subset:
  - Direct Semantics and RDF-based Semantics yield the same **(complete and sound)** results.

## OWL 2 Entailment Regimes: Complexity and Profiles (cont.)

- **OWL 2 RL** defines a syntactic subset of OWL 2. For RDF graphs that fall into this syntactic subset:
  - Direct Semantics and RDF-based Semantics yield the same (**complete and sound**) results.
  - For Direct Semantics the input RDF graph has to satisfy some constraints.

## OWL 2 Entailment Regimes: Complexity and Profiles (cont.)

- **OWL 2 RL** defines a syntactic subset of OWL 2. For RDF graphs that fall into this syntactic subset:
  - Direct Semantics and RDF-based Semantics yield the same (**complete and sound**) results.
  - For Direct Semantics the input RDF graph has to satisfy some constraints.
  - The RDF-Based semantics can be use with any RDF graph.

---

# Laboratory Session

# Laboratory

- Small exercise about OWL 2 RL entailment (very similar to RDFS Semantics).
- SPARQL 1.1 queries
- We are using OWL 2 RL reasoning (or similar in Jena). What if the modelled ontology is not in this profile?
- About lab 6 solution.
- Global picture.

# 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>