

Knowledge Graphs

Lecture 11: Ontology-Based Query Answering

Markus Krötzsch

Knowledge-Based Systems

TU Dresden, 30 Jan 2025

More recent versions of this slide deck might be available.
For the most current version of this course, see
https://iccl.inf.tu-dresden.de/web/Knowledge_Graphs/en

Review

OWL and Knowledge Graphs

OWL and RDF

RDF and OWL are intended to be widely compatible.

OWL property assertions correspond to RDF triples:

OWL axiom (FSS)

RDF triples (Turtle)

ObjectPropertyAssertion($P \ e \ f$) $e \ P \ f$.

DataPropertyAssertion($P \ e \ \ell$) $e \ P \ \ell$.

Simple ontological statements can be expressed with special properties, e.g.:

OWL axiom (FSS)

RDF triples (Turtle)

ClassAssertion($C \ e$)

$e \ \text{rdf:type} \ C$

SubClassOf($C \ D$)

$C \ \text{rdfs:subClassOf} \ D$

EquivalentClasses($C \ D$)

$C \ \text{owl:equivalentClass} \ D$

SubObjectPropertyOf($P \ Q$)

$P \ \text{rdfs:subPropertyOf} \ Q$

EquivalentObjectProperties($P \ Q$)

$P \ \text{owl:equivalentProperty} \ Q$

OWL to RDF mapping

- Some further OWL axioms can be expressed with single triples,
- but most OWL expressions and axioms require multiple triples

Example 11.1: The axiom

```
EquivalentClasses(  
  eg:MultinationalOrganisation  
  ObjectMinCardinality( 2 ObjectInverseOf( eg:memberOf ) eg:Country )  
)
```

can be represented in RDF by adding auxiliary nodes for each OWL expression:

```
eg:MultinationalOrganisation owl:equivalentClass [  
  rdf:type owl:Restriction ;  
  owl:minQualifiedCardinality 2 ;  
  owl:onProperty [ owl:inverseOf eg:memberOf ] ;  
  owl:onClass eg:Country  
] .
```

Summary: OWL and RDF

RDF-based syntax is easy:

- OWL ontologies can be represented in RDF
- Basic assertions about individuals just become RDF triples; more complex axioms are mapped to small (tree-like) graphs to capture the OWL syntax
- Harder to read, write, and parse than FSS or Manchester Syntax

RDF-based semantics is tricky:

- RDF tradition is that semantic conclusions are part of the graph
 $\leadsto$ possibly dangerous interplay between syntax and semantics
- A non-contradicting **RDF-based semantics** has been defined in OWL 2, but it often has fewer conclusions than one might expect (and sometimes more)
- We use OWL's **Direct Semantics**, where RDF is just syntax and the semantics is based on sets

OWL and Knowledge Graphs (1)

Ontology-Based Query Answering (OBQA):

- Knowledge graph = assertions about individuals (including their `rdf:type`)
- Ontology = terminological information based on KG vocabulary
- Entailed OWL assertions = additional KG facts that can be deduced

↪ Read information from “virtual” knowledge graph that consists of all entailed facts, rather than just the ones that are given

OWL and Knowledge Graphs (1)

Ontology-Based Query Answering (OBQA):

- Knowledge graph = assertions about individuals (including their `rdf:type`)
- Ontology = terminological information based on KG vocabulary
- Entailed OWL assertions = additional KG facts that can be deduced

↪ Read information from “virtual” knowledge graph that consists of all entailed facts, rather than just the ones that are given

Possible benefits

- Reduce redundancy in KG part
- Ensure systematic treatment of data
- Enrich vocabulary with derived concepts

Possible challenges

- OWL modelling requires some expertise
- Requires systems that (re)compute entailments
- Wrong/unexpected conclusions may be harder to fix

OWL and Knowledge Graphs (2)

There are some other possible uses of OWL with KGs.

Ontology-Based Data Access (OBDA)

- KG is constructed from relational database using mapping rules
- OWL used like in OBQA on constructed RDF

~> KG access on legacy RDBMS

Vocabulary documentation and schema

- OWL used to describe (intended) meaning of KG vocabulary
- No use in query answering, but as formal “schema”

~> possible use, e.g., to generate SPARQL queries from text

Data integration

- OWL ontologies can also describe relationships between the vocabularies of two separate knowledge graphs

~> can help in exchanging data or federating queries

OWL Reasoning

Main OWL Reasoning Task

Definition 11.2: The standard reasoning tasks of OWL are:

- **Axiom entailment:** Is a given OWL axiom A a logical consequence of a given OWL ontology O ? Special cases of this are **subsumption checking** (asking for SubClassOf axioms) and **instance retrieval** (asking for ClassAssertion axioms).
- **Ontology entailment:** Are all axioms in ontology O_{out} logical consequences of ontology O_{in} ?
- **Consistency:** Is the ontology O logically satisfiable (i.e., does it have a model)?

¹For most cases, this is studied in description logics; a few OWL axioms need special care.

Main OWL Reasoning Task

Definition 11.2: The standard reasoning tasks of OWL are:

- **Axiom entailment:** Is a given OWL axiom A a logical consequence of a given OWL ontology O ? Special cases of this are **subsumption checking** (asking for SubClassOf axioms) and **instance retrieval** (asking for ClassAssertion axioms).
- **Ontology entailment:** Are all axioms in ontology O_{out} logical consequences of ontology O_{in} ?
- **Consistency:** Is the ontology O logically satisfiable (i.e., does it have a model)?

The standard reasoning tasks are equivalent in the sense that an algorithm for one of them can be used to solve the others.¹

Example 11.3: An ontology is consistent exactly if it does not entail any contradictory axioms (such as SubClassOf(owl:Thing owl:Nothing)).

¹For most cases, this is studied in description logics; a few OWL axioms need special care.

Complexity of OWL Reasoning

For the Direct Semantics (based on sets and first-order logic),
all standard reasoning tasks are **decidable**, but **of rather high complexity**:

- OWL 2 reasoning in general is N2ExpTime-complete (combined complexity) and NP-complete (data complexity¹)
- Combined complexity drops to NExpTime-complete if certain uncommon uses of property chains are excluded.
- Combined complexity remains ExpTime-complete even if only a few features are allowed (e.g., with intersection, all values, subclasses).

~> unlikely to work at large scales

¹For OWL ontologies, “data” refers to the class and property assertions.

Complexity of OWL Reasoning

For the Direct Semantics (based on sets and first-order logic),
all standard reasoning tasks are **decidable**, but **of rather high complexity**:

- OWL 2 reasoning in general is N2ExpTime-complete (combined complexity) and NP-complete (data complexity¹)
- Combined complexity drops to NExpTime-complete if certain uncommon uses of property chains are excluded.
- Combined complexity remains ExpTime-complete even if only a few features are allowed (e.g., with intersection, all values, subclasses).

~> unlikely to work at large scales

Nevertheless, OWL 2 reasoning is possible in practice:

- Efficient reasoners exist, e.g., free systems HermiT, Konklude, or Pellet
- Typical OWL ontologies are much smaller than KGs (100s–100,000s of axioms)
- Worst-case complexity not typical for practical use

¹For OWL ontologies, “data” refers to the class and property assertions.

OWL for KGs?

OWL 2 is still too complex for some use cases, including OBQA with large KGs.

OWL for KGs?

OWL 2 is still too complex for some use cases, including OBQA with large KGs.

To address this issue, several **OWL profiles** (=sublanguages) have been standardised:

- **OWL EL:** “Existential language” focussed on axioms with intersection and some values, but without union, negation, number restrictions, or all values.
- **OWL RL:** “Rule language” allowing only those OWL axioms that can be written as Datalog rules
- **OWL QL:** “Query language” that allows OWL ontologies that can be compiled into SPARQL queries

OWL for KGs?

OWL 2 is still too complex for some use cases, including OBQA with large KGs.

To address this issue, several **OWL profiles** (=sublanguages) have been standardised:

- **OWL EL:** “Existential language” focussed on axioms with intersection and some values, but without union, negation, number restrictions, or all values.
- **OWL RL:** “Rule language” allowing only those OWL axioms that can be written as Datalog rules
- **OWL QL:** “Query language” that allows OWL ontologies that can be compiled into SPARQL queries

~> OWL EL is most commonly used for modelling terminologies

(typically without any data), especially in life sciences and medical

~> OWL RL and OWL QL each are suited for ontology-based query answering over KGs

Reasoning in OWL RL

OWL RL imposes **syntactic restrictions** that allow axioms to be expressed as rules:

- Distinction of subclasses (rule bodies) and superclasses (rule heads)
- Some features, e.g., some values and union, only in subclasses
- Some features, e.g., all values and negation, only in superclasses

Reasoning over KGs can be implemented as **special case of Datalog reasoning**:

- Entailments are typically computed “bottom-up” (applying rules to what is given, adding new conclusions)
- Results in “materialised” knowledge base, with all entailments made explicit

Note: This reasoning approach separates terminological OWL axioms (used as rules) and assertions (used as facts)

OWL RL: Example

Example 11.4: The OWL RL axiom

```
SubClassOf(  
  ObjectUnionOf(  
    ObjectHasValue( eg:occupation eg:composer )  
    ObjectSomeValuesFrom( eg:hasComposed eg:Music )  
  )  
  eg:Composer  
)
```

corresponds to the rules

$$\text{occupation}(X, \text{composer}) \rightarrow \text{Composer}(X)$$
$$\text{hasComposed}(X, Y) \wedge \text{Music}(Y) \rightarrow \text{Composer}(X)$$

Reasoning in OWL QL

OWL QL imposes **syntactic restrictions** that allow axioms to be expressed in queries:

- Distinction of subclasses and superclasses
- In most cases, even more restrictive than OWL RL and OWL EL
- Besides union, also intersection is now excluded; no transitivity or property chains either

Reasoning over KGs can be implemented by **query rewriting**:

- A given query (Basic Graph Pattern) is “enriched” with ontological information to cover all cases where a match for the pattern would follow from the ontology
- Results in a rewritten query that can be answered by a usual database system to get all entailments

Note: The standard rewriting approach separates terminological OWL axioms (used for rewriting) and assertions (managed by database)

OWL QL: Example

Example 11.5: Consider the OWL QL ontology

```
SubClassOf( ObjectSomeValuesFrom( eg:hasComposed owl:Thing )  
           eg:Composer )
```

```
EquivalentClasses( eg:Author eg:Writer )
```

The SPARQL query

```
SELECT DISTINCT ?X WHERE { ?X rdf:type eg:Composer, eg:Author . }
```

can then be rewritten to the following:

```
SELECT DISTINCT ?X WHERE {  
  ?X rdf:type eg:Composer, eg:Author . UNION  
  ?X rdf:type eg:Composer, eg:Writer . UNION  
  ?X eg:hasComposed []; rdf:type eg:Author . UNION  
  ?X eg:hasComposed []; rdf:type eg:Writer .  
}
```

Limits of OWL

Tree-shaped patterns

An important structural restriction of OWL is that only **tree-shaped patterns** can be modelled.

Tree-shaped: When expressing the pattern as a SPARQL BGP, the triples form a tree structure.

Example 11.6: The OWL class expression

```
ObjectIntersectionOf(  
  ObjectSomeValuesFrom( eg:father ObjectSomeValuesFrom( eg:spouse owl:Thing ) )  
  ObjectSomeValuesFrom( eg:mother ObjectSomeValuesFrom( eg:spouse owl:Thing ) )  
)
```

describes the class of things that have a father and a mother who each are married. In SPARQL, we could express this in a tree-shaped BGP (with root ?X):

```
SELECT ?X WHERE {  
  ?X eg:father ?F. ?F eg:spouse ?SF .  
  ?X eg:mother ?M. ?M eg:spouse ?SM .  
}
```

Restrictions from the tree shape

SPARQL and Datalog are not restricted to trees.

Example 11.7: We can query for individuals whose parents are married to each other in SPARQL

```
SELECT ?X WHERE {  
    ?X eg:father ?F. ?F eg:spouse ?M .  
    ?X eg:mother ?M. ?M eg:spouse ?F .  
}
```

and in Datalog

```
result(?x) :- father(?x,?f), mother(?x,?m), spouse(?m,?f),  
              spouse(?f,?m) .
```

It is impossible to capture this in an OWL class expression.

However, terminological entailment (e.g., class subsumption or query equivalence) for Datalog and SPARQL is **undecidable**.

Summary

OWL can be used with **RDF-based knowledge graphs** and other databases (**OBQA**, **OBDA**)

Reasoning in OWL is **decidable but very complex**

OWL RL and **OWL QL** are **profiles** of OWL that support reasoning on large knowledge graphs (but tools that realise this scalability in practice remain rare)

What's next?

- Constraints for knowledge graphs
- Consultation
- Examinations