Technische Universität Dresden

International Center for Computational Logic

Knowledge Graphs

Lecture 10: Ontologies for Knowledge Graphs

Markus Krötzsch

Knowledge-Based Systems

TU Dresden, 6 Jan 2026

Review

A First Introduction to OWL

Enriching knowledge graphs with ontological models

Example 10.1: In Wikidata, find everybody who is a composer by occupation or who has composed something:

```
SELECT ?person
WHERE {
  { ?person wdt:P106 wd:Q36834 } # ?person occupation: composer
  UNION
  { ?music wdt:P86 ?person } # ?music composer: ?person
}
```

Shouldn't the knowledge graph "know" what it means to be a composer in Wikidata?

⇝ Encoding of conceptual knowledge like "What is a composer?" requires ontological modelling

What is an Ontology?

"An ontology is a formal, explicit specification of a shared conceptualisation."[1]

- Conceptualisation: the terms, concepts, and relations that are of interest
  - shared: a group has agreed on this; relevant for communication
- Specification: an unambiguous, detailed account
  - formal: referring to mathematical, well-defined foundations
  - explicit: written up in a definite form

Common further and implicit assumptions:

- Ontologies are about a domain of interest, and valid mainly in this context
- They have an abstract, mathematical part with a logical semantics (ontology=logical theory)
- They have a concrete, non-mathematical part that is documented carefully (ontology=structured terminology)

[1] Studer, Benjamins, & Fensel: Knowledge engineering: Principles and methods. Data & Knowl. Eng. vol 25, 1998

Ontologies, schemas, taxonomies, and terminologies

Various related notions exist:

- Schema: information about the structure of information encoding (database schema, XML schema, . . .)
- Terminology: a set of terms with precise definitions (often in natural language)
- Vocabulary: a set of terms specific relevance to the domain
- Classification: Organisation of objects in the domain in “classes” (sets), and their relationships
- Taxonomy: strictly hierarchical classification (like the taxonomy of species)
- Ontology: A description that may contain some or all of the above

There is no formal, generally accepted definition for any of these concepts.

The Web Ontology Language OWL

OWL is a W3C[1] standard for expressing ontologies

- Mostly covers the formal, mathematical meaning (semantics) and explicit specification (syntax) of ontologies
- First proposed in 2004 (OWL 1), updated in 2012 (OWL 2)
- Conceptually based on description logics and RDF, using ideas from both worlds
- Used with data or as stand-alone knowledge model
- Supported by other W3C standards (esp. RDF and SPARQL)

In this course: focus on ontological modelling with OWL 2 for knowledge graphs

W3C creates open standards: patent-free & freely accessible

- Gentle OWL 2 introduction: `https://www.w3.org/TR/owl2-primer/`
- Overview of all parts of the standard: `https://www.w3.org/TR/owl2-overview/`

[1] World Wide Web Consortium

A Note on OWL Syntaxes

Similar to RDF graphs, OWL ontologies can be encoded in many standard syntactic forms.

The main three approaches are:

- Functional-Style Syntax (FSS): OWL structures are written as nested function terms
- Manchester Syntax: Compact human-oriented syntax, inspired by description logics
- RDF encoding: OWL ontologies can be expressed as special RDF graphs, and any RDF syntax can be used to store these graphs

In this course: FSS as main syntax, since it is closest to the structure of OWL expressions according to the standard (it is also mirrored in typical software tools)

OWL: Basic notions

OWL-based models refer to three kinds of entities:

1. **Domain elements:** individual abstract objects and data literals
2. **Properties:** binary relationships between elements
3. **Classes:** sets of elements

$\rightsquigarrow$ the syntactic names of relevant elements, properties, and classes form the vocabulary of the ontology.

Comparison to RDF:

- Domain elements and properties are similar to the structures described in RDF graphs
- Syntactic identifiers same as in RDF (IRIs, datatype literals)
- Classes are a new concept (RDF can model class membership with property `rdf:type`, but RDF graphs do not distinguish classes)
- OWL requires stricter typing, e.g., elements cannot also be properties at the same time, properties are declared as object properties (whose values are abstract individuals) or data properties (whose values are data literals)

Modelling in OWL: Assertions about elements

OWL describes relationships between the vocabulary elements using OWL axioms.

Assertional axioms can express information about elements and their direct relations. The most important ones are:[1]

OWL Functional-Style Syntax	**Intended meaning**
ClassAssertion(C e)	Individual e is an element of class C
	Example: ClassAssertion(eg:Person eg:kim)
ObjectPropertyAssertion(P e f)	Individual e relates to individual f through object property P
	Example: ObjectPropertyAssertion(eg:likes eg:kim eg:sam)
DataPropertyAssertion(P e ℓ)	Individual e relates to literal ℓ through data property P
	Example: DataPropertyAssertion(eg:h-index eg:kim 42)

Note: Property assertions are similar to special types of triples in RDF.

[1] We assume eg: to be a declared prefix. OWL supports prefix declarations similar to Turtle.

Modelling in OWL: Assertions about elements

OWL describes relationships between the vocabulary elements using OWL axioms.

Terminological axioms express relationships of classes and properties. The most important ones are:[1]

OWL Functional-Style Syntax	**Intended meaning**
SubClassOf(C D)	Every element of class C is also an element of class D
	Example: SubClassOf(eg:Composer eg:Person)
EquivalentClasses(C D)	Class C and class D have exactly the same elements
	Example: EquivalentClasses(eg:Person eg:Human)
SubObjectPropertyOf(P Q)	All elements related by property P are also related by Q
	Example: SubObjectPropertyOf(eg:fatherOf eg:parentOf)

$\rightsquigarrow$ These can be used to express hierarchies of classes and properties.

[1] We assume eg: to be a declared prefix. OWL supports prefix declarations similar to Turtle.

Complex class descriptions (1)

The power of OWL lies in its ability to define new classes by combining existing classes, properties, and elements.

The following are expressions to describe classes based on boolean operations:

OWL Functional-Style Syntax	**Intended meaning**
ObjectIntersectionOf(C D)	The class of elements that are both in C and in D
	Example: ObjectIntersectionOf(eg:Composer eg:Conductor)
ObjectUnionOf(C D)	The class of elements that are in C, in D, or in both
	Example: ObjectUnionOf(eg:Fruit eg:Vegetable)
ObjectComplementOf(C)	The class of elements that are not in C
	Example: ObjectIntersectionOf(eg:Artist ObjectComplementOf(eg:Poet))
ObjectOneOf($e_1 \cdots e_\ell$)	The class containing exactly the elements $e_1, \ldots, e_\ell$
	Example: ObjectOneOf(eg:north eg:east eg:south eg:west)

Complex class descriptions (2)

The power of OWL lies in its ability to define new classes by combining existing classes, properties, and elements.

The following are expressions to describe classes based properties:

OWL Functional-Style Syntax	**Intended meaning**
ObjectSomeValuesFrom(P C)	Class of elements that have a P-relationship to something in C Example: ObjectSomeValuesFrom(eg:livesIn eg:EUCountry)
ObjectHasValue(P e)	Class of elements that have a P-relationship to element e Example: ObjectHasValue(eg:livesIn eg:germany)
ObjectAllValuesFrom(P C)	Class of elements that have P-relationships only to things in C Example: ObjectAllValuesFrom(eg:hasChild eg:Professor)

Note: Like $\forall$ in logic, ObjectAllValuesFrom covers the extreme case where something has no P-relation at all.

Example: Complex class expressions in axioms

Class expressions can be used in axioms to define more complex statements.

Example 10.2: "Composers are people who have occupation composer or have composed something:"

```
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
EquivalentClasses(
    eg:Composer
    ObjectUnionOf(
        ObjectHasValue( eg:occupation eg:composer )
        ObjectSomeValuesFrom( eg:hasComposed owl:Thing )
    )
)
```

owl:Thing is a reserved OWL class expression that denotes the class of all things.
Its complement is **owl:Nothing**, the empty class.

OWL ontologies

There are more features and axiom types in OWL, some of which we will see later.
For now, let us summarise:

Definition 10.3: An OWL ontology is a set of OWL axioms (or a document that is a serialisation of this information).

OWL also has some non-logical features that we have ignored in this definition:

- Headers with information about the ontology as a whole
- Prefix declarations (as seen in example)
- Declarations of vocabulary elements (e.g., Declaration(Class(eg:Composer)))
- Annotations (“comments”)
- Facilities to import the axioms of other ontologies

Some other aspects are relevant for interpreting OWL, but cannot be stated in the ontology.
An example is the set of datatypes that is supported.

OWL Semantics

OWL semantics based on first-order interpretations

The semantics of OWL is based on interpretations $\mathcal{I}$, similar to interpretations in first-order logic:

- $\Delta^{\mathcal{I}}$ is the non-empty domain of $\mathcal{I}$ (the set of all existing elements/values)[1]
- for every individual element e, $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ is some domain element
- for every literal ℓ, $\ell^{\mathcal{I}}$ is the fixed value of ℓ as defined by its datatype
- for every class C, $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ is some subset of the domain
- for every property P, $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ is some binary relation over the domain

Note 1: There are many possible interpretations of an ontology, which do not have to agree on the values $\Delta^{\mathcal{I}}$, $e^{\mathcal{I}}$, $C^{\mathcal{I}}$, and $P^{\mathcal{I}}$ above, but they must agree on literals.

Note 2: When interpreting individual elements, OWL does not make the Unique Name Assumption (UNA): $e^{\mathcal{I}}$ is allowed to be the same as $f^{\mathcal{I}}$, even if e and f are different names. Conversely, some domain elements $\delta \in \Delta^{\mathcal{I}}$ may not have any name n with $n^{\mathcal{I}} = \delta$.

Note 3: Comparing to first-order logic, classes and properties correspond to unary and binary predicates, respectively.

[1] If we want to talk about data literals, their values must also be in $\Delta^{\mathcal{I}}$.

OWL semantics based on sets: class expressions

Looking at a particular interpretation $\mathcal{I}$, the expressions defined so far correspond to relations and operations on sets.

FSS expression E	**Set-based semantics** $E^{\mathcal{I}}$
owl:Thing	$\Delta^{\mathcal{I}}$
owl:Nothing	$\emptyset$
ObjectIntersectionOf($C\ D$)	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
ObjectUnionOf($C\ D$)	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
ObjectComplementOf(C)	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
ObjectOneOf($e_1 \cdots e_\ell$)	$\{e_1^{\mathcal{I}}, \ldots, e_\ell^{\mathcal{I}}\}$
ObjectSomeValuesFrom($P\ C$)	$\{d \mid$ there is x such that $\langle d, x\rangle \in P^{\mathcal{I}}$ and $x \in C^{\mathcal{I}}\}$
ObjectHasValue($P\ e$)	$\{d \mid \langle d, e^{\mathcal{I}}\rangle \in P^{\mathcal{I}}\}$
ObjectAllValuesFrom($P\ C$)	$\{d \mid$ for all x: if $\langle d, x\rangle \in P^{\mathcal{I}}$ then $x \in C^{\mathcal{I}}\}$

OWL semantics based on sets: axioms

An axiom E is statisfied by the interpretation $\mathcal{I}$, written $\mathcal{I} \models E$, if the respective condition in the following table holds:

FSS axiom A	**Condition for** $\mathcal{I} \models A$
ClassAssertion($C\ e$)	$e^{\mathcal{I}} \in C^{\mathcal{I}}$
ObjectPropertyAssertion($P\ e\ f$)	$\langle e^{\mathcal{I}}, f^{\mathcal{I}} \rangle \in P^{\mathcal{I}}$
DataPropertyAssertion($P\ e\ \ell$)	$\langle e^{\mathcal{I}}, \ell^{\mathcal{I}} \rangle \in P^{\mathcal{I}}$
SubClassOf($C\ D$)	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
EquivalentClasses($C\ D$)	$C^{\mathcal{I}} = D^{\mathcal{I}}$
SubObjectPropertyOf($P\ Q$)	$P^{\mathcal{I}} \subseteq Q^{\mathcal{I}}$

$\mathcal{I}$ satisfies an ontology O, written $\mathcal{I} \models O$, if $\mathcal{I}$ satisfies every axiom in O.

Logical conclusions from OWL ontologies

When using OWL ontologies, the key question is
"Which logical consequences follow from the ontology?"

Definition 10.4: An OWL axiom A is a logical consequence (or conclusion or entailment) of an OWL ontology O if, for all OWL interpretations $\mathcal{I}$, if $\mathcal{I} \models O$ then $\mathcal{I} \models A$.
In this case, we also write $O \models A$.

Example 10.5: Let O be the ontology with axioms

SubClassOf(ObjectSomeValuesFrom(eg:hasComposed owl:Thing) eg:Composer)
ObjectPropertyAssertion(eg:hasComposed eg:philip eg:koyaanisqatsi)

Then $O \models$ ClassAssertion(eg:Composer eg:philip).

It is not clear from the definition how to compute conclusions – we cannot consider all possible interpretations individually –, but OWL was designed so that this task is decidable, and there are many algorithms. More on this later.

More modelling features of OWL

Different ways to say the same

OWL is a rich language, and it is often possible to express the same relationships in different ways.

Example 10.6: The axiom EquivalentClasses(eg:Person eg:Human) can equivalently be expressed with two axioms

SubClassOf(eg:Person eg:Human) and SubClassOf(eg:Human eg:Person) .

Example 10.7: The class ObjectHasValue(eg:livesIn eg:germany) can equivalently be expressed as

ObjectSomeValuesFrom(eg:livesIn ObjectOneOf(eg:germany)).

So ObjectHasValue is syntactic sugar that is not adding to the expressive power.

More syntactic sugar

OWL has a number of further axioms, that can be expressed with what we have already:

OWL Functional-Style Syntax	**Intended meaning**
EquivalentObjectProperties($P\ Q$)	Properties P and Q represent the same relation
SameIndividual($e\ f$)	Individuals e and f refer to the same domain element
DifferentIndividuals($e\ f$)	Individuals e and f refer to distinct domain elements
DisjointClasses($C\ D$)	Class C and class D have no element in common
NegativeObjectPropertyAssertion($P\ e\ f$)	Elements e and f are not related by property P

In fact, all assertional axioms we presented earlier (class and property assertions) can also be written using subclass. (More about these cases in the exercise.)

Why so much sugar? Syntactic sugar simplifies syntax for common special cases. It can also help tools to see that a case is special, which may simplify computations (for example, it is easier to reason with property assertions than with general subclass axioms).

More features that are not just sugar

Further class expressions that really add expressivity:

OWL Functional-Style Syntax	**Intended meaning**
ObjectMaxCardinality($n\ P\ C$)	Things that have P relations to at most n distinct elements
ObjectMinCardinality($n\ P\ C$)	Things that have P relations to at least n distinct elements
ObjectHasSelf(P)	Things that have a P relation to themselves

Further property expressions that really add expressivity:

OWL Functional-Style Syntax	**Intended meaning**
ObjectInverseOf(P)	The inverse (reverse) of the relation P

While there are many class expressions, ObjectInverseOf is the only property expression in OWL.

Example: Further complex axioms

The additional features can be used to make further types of statements.

Example 10.8: "A multinational organisation is one that has at least two countries as members."

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

Example 10.9: "The child relation is the inverse of the parent relation."

```
EquivalentObjectProperties( eg:hasChild  ObjectInverseOf( eg:hasParent ) )
```

Note: It is a common error to confuse the direction of properties. Hence 'hasParent' and 'parentOf' might be better names than 'parent'.

Axioms about properties

OWL has some special axioms to describe properties:

OWL Functional-Style Syntax	**Intended meaning**
SymmetricObjectProperty(P)	P is a symmetric relation, i.e., is its own inverse
AsymmetricObjectProperty(P)	P is assymmetric, i.e., no pair of elements relates both ways
DisjointObjectProperties(P Q)	P and Q have no pairs of elements in common
IrreflexiveObjectProperty(P)	no element is P-related to itself
FunctionalObjectProperty(P)	every element can have at most one P value
TransitiveObjectProperty(P)	P is a transitive relation

Some of them are syntactic sugar (exercise!).

Property chains

Transitivity can be generalised to arbitrary property chains, which can only be used in as the subproperty in the following form of axiom:

SubObjectPropertyOf(ObjectPropertyChain(P_1 P_2) Q)

This axiom is satisfied by interpretation $\mathcal{I}$ if

for all elements $\delta_1, \delta_2, \delta_3 \in \Delta^{\mathcal{I}}$, if $\langle \delta_1, \delta_2 \rangle \in P_1^{\mathcal{I}}$ and $\langle \delta_2, \delta_3 \rangle \in P_2^{\mathcal{I}}$, then $\langle \delta_1, \delta_3 \rangle \in Q^{\mathcal{I}}$.

Example 10.10: "Sisters of parents are aunts."

```
SubObjectPropertyOf(
    ObjectPropertyChain( eg:hasParent eg:hasSister )
    eg:hasAunt
)
```

Example 10.11: Transitivity is a special case of property chains with $P_1 = P_2 = Q$.

Property chains: Terms and Conditions

OWL imposes several limitations on the use of property chains in order to keep reasoning decidable for basic cases.

1. **Simple properties:** Direct or indirect super-properties of property chains are non-simple, other properties simple. Only simple properties are allowed in cardinality restrictions (and some other places).
2. **Regularity:** Property chain axioms, especially when they are recursive, can express patterns along property paths. OWL requires that these patterns form a regular language (through some technical conditions on the syntax).

Summary

Ontologies are used to express structural knowledge that is more general than the concrete data in knowledge graphs

The Web Ontology Language OWL is a W3C standard

OWL has many types of axioms and expressions to describe complex relationships between individuals, properties, and classes

The set-based semantics of OWL is similar to the semantics of first-order logic

What's next?

- Computing OWL deductions
- Ontology-based query answering
- Constraints for knowledge graphs