COMP3220:
Document Processing and Semantic Technologies
OWL 2 and Profiles

Rolf Schwitter

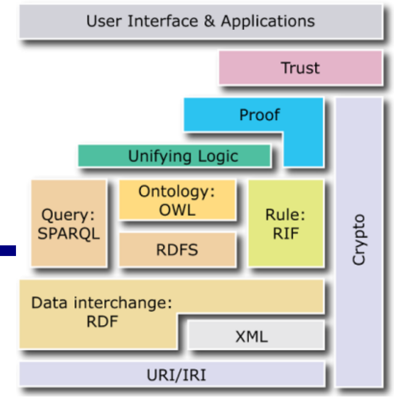
Rolf.Schwitter@mq.edu.au

Today’s Agenda

- What is OWL 2?
- OWL Full versus OWL DL
- Reasoning Capabilities
- Summary of OWL 2 Constructors and Axioms
- OWL 2 Profiles

© Macquarie University 2022

What is OWL 2?



- OWL 2 (Web Ontology Language) is the ontology language of the Semantic Web.
- OWL 2 Full is a straightforward extension of RDF(S).
- OWL 2 Full is undecidable.
- OWL 2 DL is a subset of OWL Full.
- OWL 2 DL is based on a version of Description Logic (DL).
- OWL 2 DL is decidable.
- OWL 2 DL subsumes three profiles:
  - OWL 2 EL, OWL 2 QL and OWL 2 RL.

OWL Full versus OWL DL

- OWL Full = OWL RDF-based Semantics:
  - all constructors can be used in an unrestricted way
  - reasoning works with any RDF document
  - depending on the input, reasoning might not terminate.
- OWL DL = OWL Direct Semantics:
  - based on a version of Description Logic
  - accepts only certain well-formed RDF documents
  - makes restrictions on the use of constructors
  - guarantees termination.

© Macquarie University 2022

Compatibility of OWL and RDF(S)

- Ideally, OWL would be an extension of RDF(S) and add additional constructors to support richer expressiveness.
- But **`rdfs:Class`** and **`rdf:Property`** are powerful constructors.
- These constructors lead to uncontrollable computational properties if the logic is combined with other constructors:

```
:Eagle rdf:type rdfs:Class .  % Eagle is a class.
:harry rdf:type :Eagle .
:Eagle rdf:type :Species .    % Eagle is an instance.
```

- Thus, we need a compromise: OWL 2 DL is such a compromise; DL stands for Description Logic.

Terminological Differences: DL versus OWL 2

- Description Logics speak about:
  - concepts
  - roles
  - individuals.
- OWL 2 speaks about:
  - classes
  - properties (object properties and data properties)
  - individuals.
- concepts = classes; roles = properties.

© Macquarie University 2022

An OWL Ontology

- An OWL travel ontology:
  https://web.science.mq.edu.au/~rolfs/teaching/travel.owl
  - contains classes:

```
:Adventure rdf:type owl:Class ;
           rdfs:subClassOf  :Activity ;
           owl:disjointWith :Relaxation ,
                            :Sightseeing ,
                            :Sports .
```

  - contains object properties:

```
:hasAccommodation rdf:type owl:ObjectProperty ;
                  rdfs:domain :Destination ;
                  rdfs:range  :Accommodation .
```

© Macquarie University 2022

An OWL Ontology

– contains data properties:

```
:hasZipCode rdf:type owl:DatatypeProperty ;
                     owl:FunctionalProperty ;
                     rdfs:domain :Contact ;
                     rdfs:range xsd:int .
```

– contains individuals:

```
:BlueMountains rdf:type owl:NamedIndividual ; :NationalPark .
:BondiBeach rdf:type owl:NamedIndividual ; :Beach .
:Cairns rdf:type owl:NamedIndividual ; :City .
```

– Contains general axioms:

```
[ rdf:type owl:AllDifferent ;
  owl:distinctMembers ( :OneStarRating :TwoStarRating )] .
```

© Macquarie University 2022

OWL 2 Syntaxes

Name of Syntax	Specification	Status	Purpose
RDF/XML	Mapping to RDF Graphs, RDF/XML	Mandatory	Interchange (can be written and read by all conformant OWL 2 software)
OWL/XML	XML Serialization	Optional	Easier to process using XML tools
Functional Syntax	Structural Specification	Optional	Easier to see the formal structure of ontologies
Manchester Syntax	Manchester Syntax	Optional	Easier to read/write DL Ontologies
Turtle	Mapping to RDF Graphs, Turtle	Optional, Not from OWL-WG	Easier to read/write RDF triples

http://www.w3.org/TR/owl2-overview/

© Macquarie University 2022

OWL 2: Concept/Class Constructors

OWL Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	$\neg$Male
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} $\sqcup$ {mary}
allValuesFrom	$\forall P.C$	$\forall$hasChild.Doctor
someValuesFrom	$\exists P.C$	$\exists$hasChild.Lawyer
maxCardinality	$\leqslant nP$	$\leqslant$1hasChild
minCardinality	$\geqslant nP$	$\geqslant$2hasChild

OWL 2: Ontology Axioms

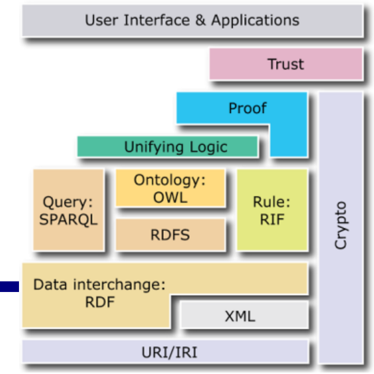
TBox

OWL Syntax	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor

ABox

OWL Syntax	DL Syntax	Example
type	$a : C$	John : Happy-Father
property	$\langle a, b \rangle : R$	$\langle$John, Mary$\rangle$: has-child

An OWL 2 Ontology in Turtle



- Specifying the prefixes:

```
@prefix : <http://example.com/owl/families/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

An OWL 2 Ontology in Turtle: TBox

- Specifying classes and properties:

```
:Person  rdf:type owl:Class .
:hasWife rdf:type owl:ObjectProperty .
:hasAge  rdf:type owl:DatatypeProperty .
```

© Macquarie University 2022

An OWL 2 Ontology in Turtle: TBox

- Specifying domains and ranges:

```
:hasAge  rdfs:domain  :Person ;
         rdfs:range   xsd:nonNegativeInteger .
```

© Macquarie University 2022

An OWL 2 Ontology in Turtle: TBox

- Specifying subclasses and subproperties:

```
:hasWife rdfs:subPropertyOf :hasSpouse .
:Mother  rdfs:subClassOf    :Woman .
```

An OWL 2 Ontology in Turtle: TBox

- Describing properties in more detail:

```
:hasParent     owl:inverseOf :hasChild .
:hasSpouse     rdf:type      owl:SymmetricProperty .
:hasChild      rdf:type      owl:AsymmetricProperty .
:hasParent     owl:propertyDisjointWith  :hasSpouse .
:hasRelative   rdf:type      owl:ReflexiveProperty .
:parentOf      rdf:type      owl:IrreflexiveProperty .
:hasHusband    rdf:type      owl:FunctionalProperty .
:hasAncestor   rdf:type      owl:TransitiveProperty .
:hasGrandparent  owl:propertyChainAxiom
                 ( :hasParent  :hasParent ) .
```

© Macquarie University 2022

An OWL Ontology in Turtle: ABox

- Adding some ABox statements:

```
:John  rdf:type             owl:NamedIndividual .
:John  rdf:type             :Father .
:John  :hasWife             :Mary .
:John  owl:differentFrom :Bill .
:John  :hasAge              51 .
```

RDF(S)

- In RDF(S) we can for example declare:
  - classes like `Country`, `Person` and `Student`
  - that `Student` is a subclass of `Person`
  - that `Australia` and `India` are both instances of `Country`
  - that `hasAge` is a property with `Person` as its domain and integer as its range
  - that `Peter` is an instance of the class `Australian` and that he `hasAge` of `48`.

OWL 2 DL

- In OWL 2 DL, we can for example additionally declare:
  - that `Country` and `Person` are disjoint classes
  - that `Australia` and `India` are distinct individuals
  - `hasCitizen` as inverse property of `hasNationality`
  - that the class `Stateless` is precisely defined as those members of the class `Person` that have no values for the property `hasNationality`
  - that the class `MultipleNationals` is precisely defined as those members of the class `Person` that have at least `2` values for `hasNationality`
  - that the class `Australian` is precisely defined as those members of the class `Person` that have `Australia` as a value of `hasNationality`
  - that `hasAge` is a functional property.

RDF(S) Reasoning Capabilities

Example 1:

Type inheritance: `rdfs:subClassOf`

```
:Morris rdf:type :Cat .
:Cat rdfs:subClassOf :Mammal .
```

implies:

```
:Morris rdf:type :Mammal .
```

© Macquarie University 2022

RDF(S) Reasoning Capabilities

Example 2:

Type inference through `rdfs:domain` and `rdfs:range`:

```
rdfs:domain :teaches :Teacher .
rdfs:range :teaches :Student .
:Bob :teaches :Mary .
```

implies:

```
:Bob rdf:type :Teacher .
:Mary rdf:type :Student .
```

© Macquarie University 2022

RDF(S) Reasoning Capabilities

Example 3:

Transitivity of `subClassOf`:

```
:Dog rdfs:subClassOf :Mammal .
:Mammal rdfs:subClassOf :Animal .
```

implies:

```
:Dog rdfs:subClassOf :Animal .
```

RDF(S) Reasoning Capabilities

Example 4:

Transitivity of `subPropertyOf`:

```
:parentOf rdfs:subPropertyOf :ancestorOf .
:ancestorOf rdfs:subPropertyOf :relativeOf .
```

implies:

```
:parentOf rdfs:subPropertyOf :relativeOf .
```

© Macquarie University 2022

OWL 2 DL Reasoning Capabilities

Example 1:

Enforcing transitivity of **`owl:TransitiveProperty`**:

```
:ancestorOf rdf:type owl:TransitiveProperty .
:Sue ancestorOf :Mary .
:Mary ancestorOf :Anne .
```

implies:

```
:Sue ancestorOf :Anne .
```

© Macquarie University 2022

OWL 2 DL Reasoning Capabilities

Example 2:

Reasoning with `owl:inverseOf`:

```
:parentOf owl:inverseOf :hasParent .
:Yolande :parentOf :Rona .
```

implies:

```
:Rona hasParent :Yolande .
```

© Macquarie University 2022

OWL 2 DL Reasoning Capabilities

Example 3:

Inheritance of disjointness constraints:

```
:Plant owl:disjointWith :Animal .
:Mammal owl:subClassOf :Animal .
```

implies:

```
:Plant owl:disjointWith :Mammal .
```

© Macquarie University 2022

OWL 2 DL Reasoning Capabilities

Example 4:

Subclasses are disjoint with the class’s complement:

```
:Animal owl:complementOf :NonAnimals .
:Mammal rdfs:subClassOf :Animal .
```

implies:

```
:Mammal owl:disjointWith NonAnimals .
```

© Macquarie University 2022

OWL 2 DL Reasoning Capabilities

Example 5:

Inferring `owl:sameAs` via `owl:FunctionalProperty`:

```
:hasMother rdf:type owl:FunctionalProperty .
:Yolande :hasMother :Margaret .
:Yolande :hasMother :Maggie .
```

implies:

```
:Margaret owl:sameAs :Maggie .
```

© Macquarie University 2022

OWL 2 DL Profiles

- OWL 2 DL is decidable but computationally hard.
- OWL 2 DL is not scalable enough for many applications.
- Therefore, the W3C defined three different tractable profiles for particular application domains:
  - OWL 2 EL: polynomial time reasoning for ontologies with a large conceptual part.
  - OWL 2 QL: fast logspace query answering for large datasets stored in RDBs.
  - OWL 2 RL: polynomial time reasoning that can be implemented using rule-based reasoning systems.

© Macquarie University 2022

OWL 2 EL

- Tailored for applications that contain very large numbers of properties and/or classes.
- Allows existential quantification.
- Does not allow disjunction and universal quantification.
- Satisfiability, subsumption, classification, and instance checking can be decided in polynomial time.
- Good for modelling large life science ontologies.
- Example: SNOMED-CT, an ontology of clinical terms with over 350,000 concepts (= classes).
  https://www.snomed.org/snomed-ct/five-step-briefing

© Macquarie University 2022

OWL 2 EL: Example in Functional-Style Syntax

```
SubClassOf(
  :Father
  ObjectIntersectionOf( :Man :Parent )
)

EquivalentClasses(
  :Parent
  ObjectSomeValuesFrom(
    :hasChild
    :Person
  )
)

EquivalentClasses(
  :NarcisticPerson
  ObjectHasSelf( :loves )
)

DisjointClasses(
  :Mother
  :Father
  :YoungChild
)

SubObjectPropertyOf(
  ObjectPropertyChain( :hasFather :hasBrother )
  :hasUncle
)

NegativeObjectPropertyAssertion(
  :hasDaughter
  :Bill
  :Susan
)
```

OWL 2 EL: Example in Turtle Syntax

```
:Father  rdfs:subClassOf  [
  rdf:type             owl:Class ;
  owl:intersectionOf  ( :Man  :Parent )
] .

:Parent  owl:equivalentClass  [
  rdf:type             owl:Restriction ;
  owl:onProperty       :hasChild ;
  owl:someValuesFrom   :Person
] .

:NarcisticPerson  owl:equivalentClass  [
  rdf:type         owl:Restriction ;
  owl:onProperty   :loves ;
  owl:hasSelf      true
] .

[] rdf:type     owl:AllDisjointClasses ;
   owl:members  ( :Mother  :Father :YoungChild ) .

:hasUncle  owl:propertyChainAxiom  ( :hasFather  :hasBrother ) .

[]  rdf:type                owl:NegativePropertyAssertion ;
    owl:sourceIndividual    :Bill ;
    owl:assertionProperty   :hasDaughter ;
    owl:targetIndividual    :Susan .
```


http://www.w3.org/TR/owl2-primer/

OWL 2 QL

- Tailored for applications that need to reason on top of very large volumes of data.
- Reasoning is translated into queries on databases.
- Allows subclass axioms, inverse object properties.
- Does not allow existential and universal quantification.
- This profile provides many of the main features
  - to express conceptual models such as UML class diagrams and ER diagrams
  - to define hierarchies between classes and properties.

© Macquarie University 2022

OWL 2 QL: Example in Functional-Style Syntax

```
SubClassOf(
  :ChildlessPerson
  ObjectIntersectionOf(
    :Person
    ObjectComplementOf(
      ObjectSomeValuesFrom(
        ObjectInverseOf( :hasParent )
        owl:Thing
      )
    )
  )
)

DisjointClasses(
  :Mother
  :Father
  :YoungChild
)

DisjointObjectProperties(
  :hasSon
  :hasDaughter
)

SubObjectPropertyOf(
  :hasFather
  :hasParent
)
```

OWL 2 QL: Example in Turtle Syntax

```
:ChildlessPerson  owl:subClassOf  [
  rdf:type            owl:Class ;
  owl:intersectionOf  ( :Person
                         [ owl:complementOf  [
                             rdf:type            owl:Restriction ;
                             owl:onProperty      [ owl:inverseOf  :hasParent ] ;
                             owl:someValuesFrom  owl:Thing
                           ]
                         ]
                       )
] .

[]  rdf:type    owl:AllDisjointClasses ;
    owl:members ( :Mother  :Father  :YoungChild ) .

:hasSon  owl:propertyDisjointWith  :hasDaughter.

:hasFather  rdfs:subPropertyOf  :hasParent.
```

http://www.w3.org/TR/owl2-primer/

OWL 2 RL

- Tailored for applications that want to describe rules in ontologies.
- Not allowed: existential quantification, union, and disjoint union.
- This profile is ideal, if you already have RDF data and you want to implement your business logic in rules (if/then).
- OWL 2 RL runs efficiently on business rule engines.
- It is basically a rule language (hence the RL).

© Macquarie University 2022

OWL RL: Example in Functional Style Syntax

```
SubClassOf(
  ObjectIntersectionOf(
    ObjectOneOf( :Mary :Bill :Meg )
    :Female
  )
  ObjectIntersectionOf(
    :Parent
    ObjectMaxCardinality( 1 :hasChild )
    ObjectAllValuesFrom( :hasChild :Female )
  )
)

DisjointClasses(
  :Mother
  :Father
  :YoungChild
)

SubObjectPropertyOf(
  ObjectPropertyChain( :hasFather :hasBrother )
  :hasUncle
)
```

The first axiom states that for each of Mary, Bill, and Meg who is female, the following holds: she is a parent with at most one child, and all her children (if she has any) are female.

OWL 2 RL: Example in Turtle Syntax

```
[]  rdf:type              owl:Class ;
    owl:intersectionOf    ( [ rdf:type   owl:Class ;
                              owl:oneOf  ( :Mary  :Bill  :Meg ) ]
                            :Female
                          ) ;
    rdfs:subClassOf       [
      rdf:type              owl:Class ;
      owl:intersectionOf    ( :Parent
                              [ rdf:type              owl:Restriction ;
                                owl:maxCardinality   "1"^^xsd:nonNegativeInteger ;
                                owl:onProperty        :hasChild ]
                              [ rdf:type             owl:Restriction ;
                                owl:onProperty       :hasChild ;
                                owl:allValuesFrom    :Female ]
                            )
    ] .

[]  rdf:type     owl:AllDisjointClasses ;
    owl:members  ( :Mother  :Father  :YoungChild ) .

:hasUncle  owl:propertyChainAxiom  ( :hasFather  :hasBrother ) .
```

Take-Home Messages

- OWL 2 is a family of knowledge representation languages.
- OWL 2 is a stronger language than RDF(S).
- The normative syntax of OWL 2 is RDF/XML.
- But the OWL 2 family contains many serialisations.
- There exist three OWL 2 profiles: EL, QL, and RL.
- Profiles are trimmed down versions of OWL 2.
- Focus of these profiles is computational tractability.
- The key question is: which constructors do we need to model the domain?

© Macquarie University 2022