Welcome to INFO216: Knowledge Graphs

Spring 2023

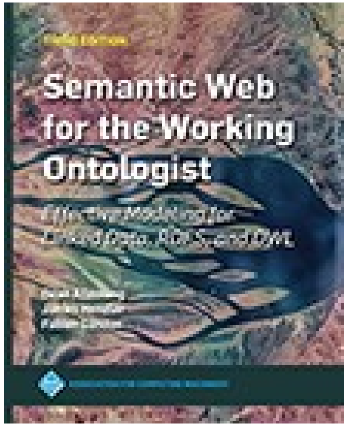
Andreas L Opdahl
<Andreas.Opdahl@uib.no>

Session 10: Formal ontologies (OWL-DL)

- Themes:
  - OWL-DL
    - core OWL concepts
    - restriction classes
  - description logic
  - decision problems

Readings

- Sources:
  - Allemang, Hendler & Gandon (2020):
    Semantic Web for the Working Ontologist, 3rd edition:
    chapters 12-13, but chapters 11-12 in the 2nd edition
  - Blumauer & Nagy (2020):
    Knowledge Graph Cookbook – Recipes that Work:
    e.g., pages 105-109, 123-124, (*supplementary*)
- Resources in the wiki <http://wiki.uib.no/info216>, e.g.:
  - OWL 2 Overview (http://www.w3.org/TR/owl-overview/)
  - OWL 2 Primer (http://www.w3.org/TR/owl-primer/):
    - show: Turtle and Manchester syntax
    - hide: other syntaxes



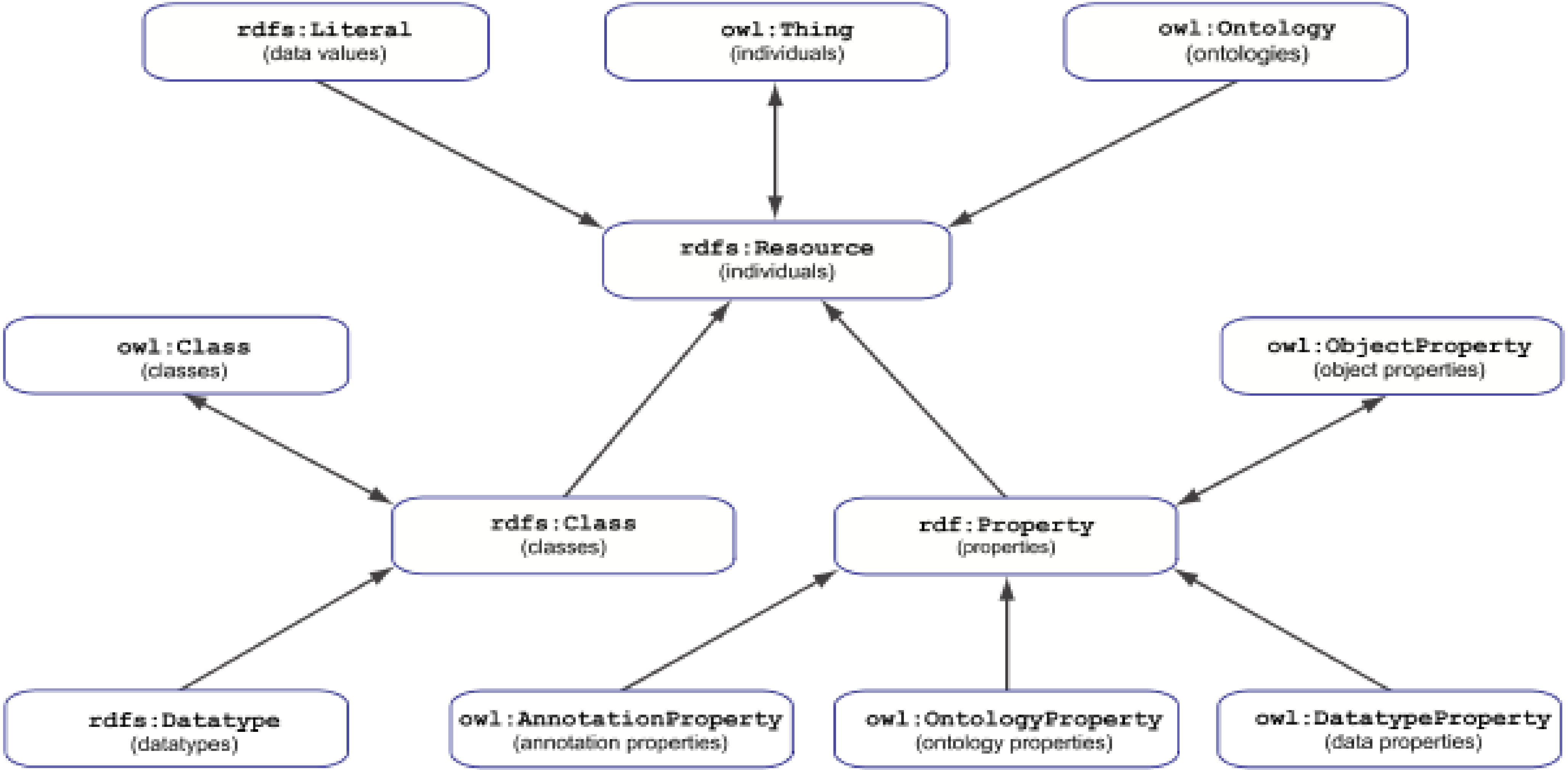
The Core
OWL Concepts

Web Ontology Language versions

- OWL "1" (2002):
  - OWL Full – anything goes
  - OWL DL – fragment of OWL Full, formal semantics through *description logic*
  - OWL Lite – simple fragment of OWL DL, not much used
- OWL 2 (2008):
  - *backwards compatible* with OWL "1"!
  - OWL2 DL – fragment of OWL2 full, extension of OWL DL
    - formal and powerful, but *reasoning can get prohibitively slow*
  - OWL2 DL – defines three faster fragments of OWL2 DL:
    - OWL2 RL – rule-based semantics, also OWL LD – for Linked Data
    - OWL2 EL – quick DL reasoning
    - OWL2 QL – suitable for query rewriting

Classes, properties, and individuals (←S08)

- Web Ontology Language (OWL):
  - builds on RDF and RDFS
  - uses classes and properties from RDF and RDFS
  - adds precision and formality
- Basic OWL-concepts:
  - owl:Thing (equivalent to rdfs:Resource)
  - owl:Class (equivalent to rdfs:Class)
  - owl:ObjectProperty (equivalent to rdf:Property)
  - owl:NamedIndividual (things with URIs and that are not classes)
- Good practice: keep *Classes, Individuals, and Properties disjoint*, i.e., no resource has more than one of them as *rdf:type*
  - *in OWL DL, this is mandatory...*



http://www.w3.org/TR/owl2-rdf-based-semantics/

Building blocks

OWL2 can be seen as
an extension of RDF and RDFS,
but can also stand on
its own feet.

- OWL 2 has three building blocks:
  - *entities:*
    - refer to real-world entities using URIs
    - owl:NamedClass, owl:NamedIndividual
    - owl:ObjectProperty, owl:DatatypeProperty, owl:AnnotationProperty, owl:ObjectProperty
  - *axioms:* ← *can be true or false!*
    - basic statements expressed by the OWL ontology
    - every triple in the RDF graph is an axiom
  - *expressions:*
    - use *constructors to*
    - *define more complex entities*
    - by *combining* simpler ones

More building blocks

- owl:Thing:
  - is equivalent to *rdfs:Resource*
  - logic interpretation: *True*
    - called the *top concept* in description logic (DL)
- owl:Nothing
  - is the empty set
  - no resource has it as its *rdf:type*
  - logic interpretation: *False*
    - called the *bottom concept* in DL

Named and constructed classes

- owl:NamedClass (with an URI):
  - semantics are given by:
    - URI-s, labels and other annotations
    - domain, range, subClassOf and other relationships
- *Constructed* (or *complex*) owl:Class:
  - built from existing classes, properties, individuals
    - which can be named *or anonymous*
  - constructed classes are *anonymous upon declaration,*
    - but can be *named* later
  - *unions, intersections* and *negations* of existing classes (←S08)
  - *enumeration* of existing individuals (←S08)
  - *restrictions* on existing properties

Object and datatype properties

- In RDF triples, the object is either a resource or a literal
  - OWL has two corresponding types of properties
  - owl:ObjectProperty:
    - rdfs:range ("verdiområde") is usually an OWL-class of individuals
    - used in axioms (e.g., RDF triples) with a *resource* object
  - owl:DatatypeProperty:
    - rdfs:range is an RDFS-datatype
    - used in axioms (e.g., RDF triples) with a *literal* object
  - the rdfs:domain ("definisjonsmengden") is always an OWL-class of individuals

Formally, owl:DatatypeProperty is rdfs:subPropertyOf owl:ObjectProperty .

Summary: core OWL concepts

- owl:Thing, owl:Nothing
  owl:NamedIndividual
- owl:NamedClass, owl:Class
- owl:ObjectProperty, owl:DatatypeProperty
- owl:AnnotationProperty, owl:OntologyProperty

Summary: more precise properties (←S08)

- owl:inverseOf
- owl:SymmetricProperty, owl:AsymmetricProperty
- owl:ReflexiveProperty, owl:IrreflexiveProperty
- owl:TransitiveProperty
- owl:FunctionalProperty, owl:InverseFunctionalProperty
- owl:hasKey
- Also:
  - negated properties
  - chained properties, e.g.:
    fam:hasGrandparent
    owl:propertyChainAxiom (:hasParent :hasParent) .

Summary: sameness and difference (←S08)

- Individuals:
  - pairwise: owl:sameAs, owl:differentFrom
  - groupwise difference: owl:AllDifferent
- Classes:
  - pairwise: owl:equivalentClass, owl:disjointWith
  - groupwise difference: owl:AllDisjointClasses
- Properties:
  - pairwise: equivalentProperty, propertyDisjointWith
  - groupwise difference: owl:AllDisjointProperties
- Membership in the groups:
  - owl:distinctMembers *(preferred)* or owl:members

Summary: complex classes (←S08)

- owl:oneOf
- owl:unionOf
- owl:intersectionOf
- owl:complementOf (and the *CWA*)
- owl:NegativePropertyAssertion, owl:sourceIndividual, owl:assertionProperty, owl:targetIndividual

OWL restriction classes

Property value restrictions

- Defining a class by a particular value on one of its properties, e.g.:
  - ex:Republican

```
        a owl:Restriction ;
        owl:onProperty dbo:hasParty ;
        owl:hasValue dbr:Republican_Party_(United_States) .
```

Property value restrictions

- Defining a class by a particular value on one of its properties, e.g.:
  - ex:Republican owl:intersectionOf (

```
ex:Republican owl:intersectionOf (
        dbr:Person
        [   a owl:Restriction ;
            owl:onProperty dbo:hasParty ;
            owl:hasValue dbr:Republican_Party_(United_States)
        ]
) .
```

Existential property restrictions

- Defining a class by the existence of a relation (object property) to an individual in (another or the same) class, e.g.:
  - ex:President owl:intersectionOf (

```
                    dbr:Person
                    [    a owl:Restriction ;
                         owl:onProperty ex:presidentOf ;
                         owl:someValuesFrom owl:Thing
                    ]
    ) .
```

- *owl:someValuesFrom:* each individual in the defined class has *at least one* object property (given by owl:onProperty) to an individual in the other class (given by owl:someValuesFrom)

Existential property restrictions

- Defining a class by the existence of a relation (object property) to an individual in (another or the same) class, e.g.:
  - ex:US_President owl:intersectionOf (

```
            dbr:Person
            [    a owl:Restriction ;
                 owl:onProperty ex:presidentOf ;
                 owl:someValuesFrom dbr:UnitedStates
            ]
) .
```

- *owl:someValuesFrom:* each individual in the defined class has *at least one* object property (given by owl:onProperty) to an individual in the other class (given by owl:someValuesFrom)

Universal property restrictions

- Defining a class by the necessity of a relation (object property) only to individuals in (another or the same) class, e.g.:
  - ex:LoyalRepublican owl:intersectionOf (

```
ex:LoyalRepublican  owl:intersectionOf (
        dbr:Person
        [     a owl:Restriction ;
              owl:onProperty dbr:hasParty ;
              owl:allValuesFrom dbr:Republican_Party_(United_States)
        ]
) .
```

Universal property restrictions

- Defining a class by the necessity of a relation (object property) only to individuals in (another or the same) class, e.g.:
  - ex:LoyalRepublican owl:intersectionOf (

```
ex:LoyalRepublican  owl:intersectionOf (
            dbr:Person
            [     a owl:Restriction ;
                  owl:onProperty dbr:hasParty ;
                  owl:allValuesFrom dbr:Republican_Party_(United_States)
            ]
    ) .
```

What is wrong here?

Universal property restrictions

- Defining a class by the necessity of a relation (object property) only to individuals in (another or the same) class, e.g.:
  - ex:LoyalRepublican owl:intersectionOf (

```
ex:LoyalRepublican  owl:intersectionOf (
        dbr:Person
        [   a owl:Restriction ;
            owl:onProperty dbr:hasParty ;
            owl:allValuesFrom dbr:Republican_Party_(United_States)
        ]
        [   a owl:Restriction ;
            owl:onProperty dbr:hasParty ;
            owl:someValuesFrom owl:Thing
        ]
) .
```

Property self-reflexion

- Defining a class by a *Self* value on one of its properties, e.g.:
  - ex:Narcissist

```
a                   owl:Restriction ;
owl:onProperty      ex:loves ;
owl:hasSelf         "true"^^xsd:boolean .
```

Datatype property restriction

- Restrictions on data range, e.g.:
  - fam:personAge rdfs:range

```
[     a rdfs:Datatype;
      owl:onDatatype  xsd:integer;
      owl:withRestrictions (
          [ xsd:minInclusive  "0"^^xsd:integer ]
          [ xsd:maxInclusive  "130"^^xsd:integer ]  )
] .
```

  - :toddlerAge rdfs:range

```
[     a rdfs:Datatype;
      owl:oneOf ( "1"^^xsd:integer  "2"^^xsd:integer)
] .
```

Cardinality restriction

- Defining a class by the number of object values its individuals have for some property, e.g.:
  - music:Quartet owl:intersectionOf (

```
music:Quartet owl:intersectionOf (
        music:Ensemble
        [   a owl:Restriction ;
            owl:onProperty music:hasInstrument ;
            owl:cardinality 4   ]
) .
```

- owl:cardinality gives the *exact cardinality*
  owl:minCardinality gives the *least cardinality*
  owl:maxCardinality gives the *greatest cardinality*

Qualified cardinality restriction

- Defining a class by the number of object values its individuals have *of a given class* for some property, e.g.:
  - pol:Triumvirate owl:intersectionOf (

```
pol:Triumvirate owl:intersectionOf (
        pol:PoliticalLeadership
        [   a owl:Restriction ;
            owl:onProperty pol:hasMember ;
            owl:qualifiedCardinality 3 ;
            owl:onClass pol:PoliticalLeader   ]
    ) .
```

- owl:qualifiedCardinality gives the *exact cardinality*
  owl:minQualifiedCardinality gives the *least cardinality*
  owl:maxQualifiedCardinality gives the *greatest cardinality*

Qualified cardinality restriction

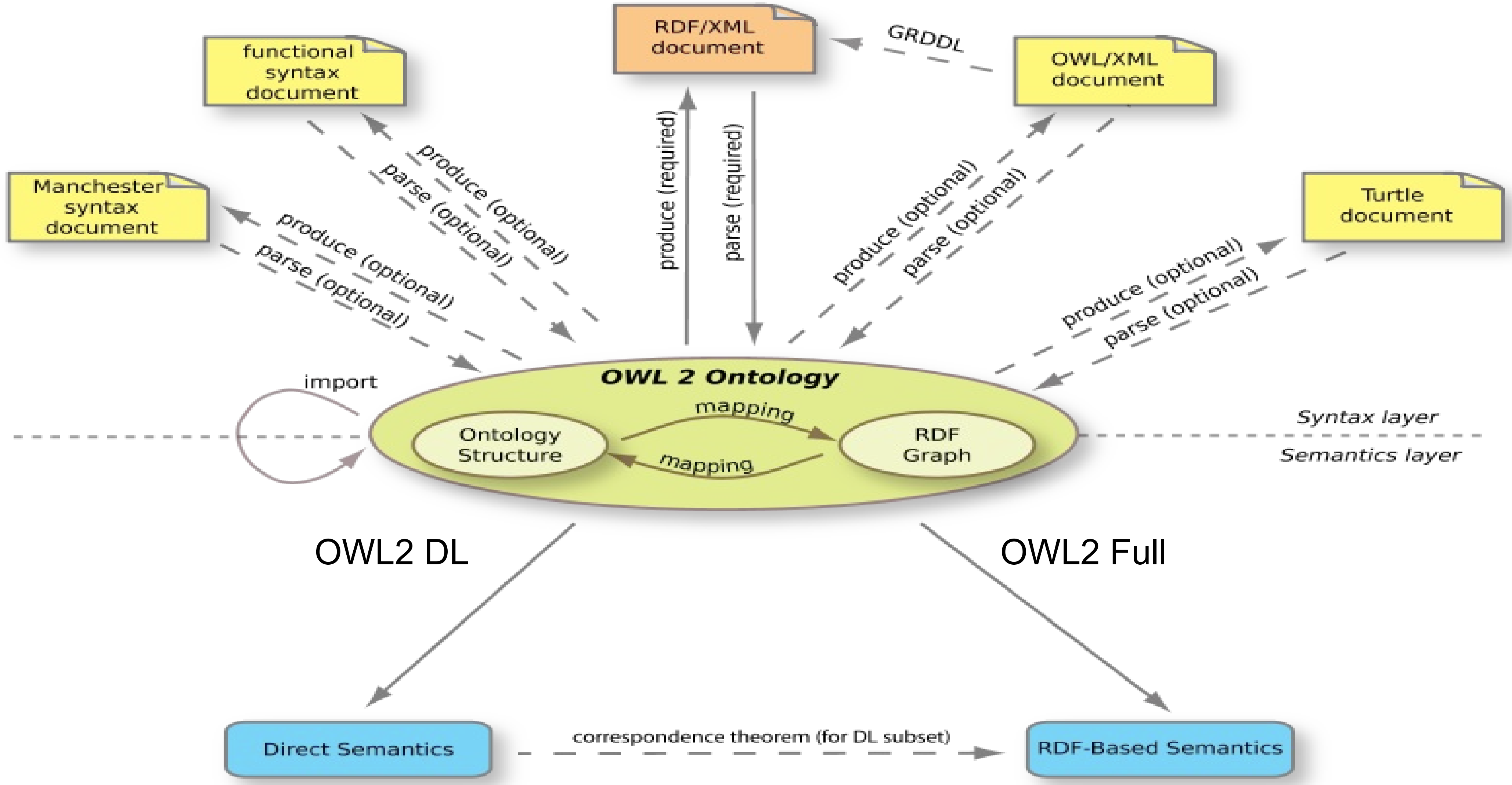
- music:StringQuartet owl:intersectionOf (

```
music:MusicalQuartet
[     a owl:Class ;
      owl:onProperty music:hasInstrument ;
      owl:qualifiedCardinality “2” ;
      owl:onClass music:Violin     ]
[     a owl:Class ;
      owl:onProperty music:hasInstrument ;
      owl:qualifiedCardinality “1” ;
      owl:onClass music:Viola      ]
[     a owl:Class ;
      owl:onProperty music:hasInstrument ;
      owl:qualifiedCardinality “1” ;
      owl:onClass music:Cello     ] ) .
```

Summary: property restrictions

- owl:Restriction owl:onProperty
- owl:someValuesFrom, owl:allValuesFrom, owl:hasValue
- owl:cardinality, owl:minCardinality, owl:maxCardinality
- owl:qualifiedCardinality, owl:minQualifiedCardinality, owl:maxQualifiedCardinality, owl:onClass

Description logic


RDF/XML document
functional syntax document
Manchester syntax document
OWL/XML document
Turtle document
GRDDL
produce (optional)
parse (optional)
produce (required)
parse (required)
import
OWL 2 Ontology
Ontology Structure
RDF Graph
mapping
mapping
Syntax layer
Semantics layer
OWL2 DL
OWL2 Full
Direct Semantics
correspondence theorem (for DL subset)
RDF-Based Semantics

Relation to OWL

- OWL DL and description logic are closely matched
  - everything in OWL DL has a DL-counterpart
  - almost everything in DL can be expressed in OWL DL
- DL is a family of logic systems:
  - some of them correspond to particular OWL profiles (more later)
  - OWL1 DL: $\mathcal{SHOIN}^{(\mathcal{D})}$
  - OWL2 DL: $\mathcal{SROIQ}^{(\mathcal{D})}$

Description logic and other logics

- *Proposition logics* are about *statements* (*propositions*):
  - **"Robin is a StudentAssistant" ⇐ "Robin is a Student" ∧ "Robin is a Teacher"**
- (First order) *predicate logics* are about *predicates* and *objects*:
  - **∀x.(StudentAssistant(x) ⇔ Student(x) ∧ Teacher(x))**
- *Description logics* are about *concepts*:
  - **StudentAssistant ≐ Student ⊓ Teacher**
    - ...and also about *roles* and *individuals*

Description logics

- Description Logic (DL)
  - a simple *fragment* of predicate logic
    - ...or, rather, a *family of such fragments*
  - not very *expressive* ("uttrykkskraftig")
  - but can answers many *decision problems* (rather) quickly
- Suitable for describing *concepts* ("begreper")
  - formal basis for *OWL DL*
  - can be used to:
    - describe *concepts* ("Tbox") and their *roles* ("Rbox")
    - describe *individuals* and their relations ("ABox")

Definition of concepts (“begreper”)

- **InternalCensor ≐ Censor ⊓ Employee**
- **ExternalCensor ≐ Censor ⊓ ¬ Employee**
- **Agent ≐ Person ⊔ Organisation ⊔ Group**
  - concepts: **InternalCensor, Censor, Employee…**
  - definition: ≐
  - conjunction (and): ⊓
  - disjunction (or): ⊔
  - negation (not): **¬**
  - nested expressions: **(   )**
- **Childless** ≐ ..using Human and Parent..

Definition of concepts ("begreper")

- **InternalCensor** $\doteq$ **Censor** $\sqcap$ **Employee**
- **ExternalCensor** $\doteq$ **Censor** $\sqcap$ $\neg$ **Employee**
- **Agent** $\doteq$ **Person** $\sqcup$ **Organisation** $\sqcup$ **Group**
  - concepts: **InternalCensor, Censor, Employee…**
  - definition: $\doteq$
  - conjunction (and): $\sqcap$
  - disjunction (or): $\sqcup$
  - negation (not): $\neg$
  - nested expressions: **(   )**
- **Childless** $\doteq$ **Human** $\sqcap$ $\neg$ **Parent**

Types of concepts ("begreper")

- **InternalCensor ≐ Censor ⊓ Employee**
- **ExternalCensor ≐ Censor ⊓ ¬ Employee**
- **Agent ≐ Person ⊔ Organisation ⊔ Group**
  - atomic (or basic, primitive) concepts:
    Censor, Employee, Person…
    - only used on the r.h.s. of definitions
  - concept expressions (complex concepts):
    Censor ⊓ Employee, ¬ Employee…
    - only used on the r.h.s. of definitions
  - defined (and named) concepts:
    InternalCensor, ExternalCensor, Agent…
    - defined on the *l.h.s. (left-hand side)* of definitions

Atomic, complex and defined concepts

- Atomic (or basic) concepts
  - given, always named
  - can only be used on the r.h.s. (right-hand side) of a $\doteq$ definition
  - correspond to simple OWL classes
- Concept expressions
  - expressed using other concepts (and roles)
  - can only be used on the r.h.s. (right-hand side) of a $\doteq$ definition
  - correspond to complex OWL classes
- Defined concepts can also be named
  - must appear on the l.h.s. (right-hand side) of a $\doteq$ definition
  - concept_name $\doteq$ concept_expression
- ...similar distinction between atomic and defined roles

Roles

- $\texttt{President} \doteq \texttt{Person} \sqcap \exists\,\texttt{presidentOf}.\top$
- $\texttt{Independent} \doteq \texttt{Person} \sqcap \neg\exists\,\texttt{hasParty}.\top$

An atomic (or basic) role

- $\texttt{USPresident} \doteq \texttt{Person} \sqcap \exists\,\texttt{presidentOf}.\texttt{UnitedStates}$
  - roles: **presidentOf, hasParty...**
  - universal concept ("top"): $\top$
  - existential restriction: $\exists$
- **Grandparent** $\doteq$ ..using Human, hasChild, Parent..
- **Grandparent** $\doteq$ ..using only Human, hasChild..
- **Uncle** $\doteq$ ..using Male, hasSibling, hasChild..

Roles

- $\mathtt{President} \doteq \mathtt{Person} \sqcap \exists\mathtt{presidentOf}.\top$
- $\mathtt{Independent} \doteq \mathtt{Person} \sqcap \neg\exists\mathtt{hasParty}.\top$
- $\mathtt{USPresident} \doteq \mathtt{Person} \sqcap \exists\mathtt{presidentOf}.\mathtt{UnitedStates}$
    - roles: **presidentOf, hasParty…**
    - universal concept ("top"): $\top$
    - existential restriction: $\exists$
- $\mathtt{Grandparent} \doteq \mathtt{Human} \sqcap \exists\mathtt{hasChild}.\mathtt{Parent}$
- **Grandparent** $\doteq$ ..using only Human, hasChild..
- **Uncle** $\doteq$ ..using Male, hasSibling, hasChild..

Roles

- $\text{President} \doteq \text{Person} \sqcap \exists\text{presidentOf}.\top$
- $\text{Independent} \doteq \text{Person} \sqcap \neg\exists\text{hasParty}.\top$
- $\text{USPresident} \doteq \text{Person} \sqcap \exists\text{presidentOf}.\text{UnitedStates}$
    - roles: **presidentOf, hasParty…**
    - universal concept ("top"): $\top$
    - existential restriction: $\exists$
- $\text{Grandparent} \doteq \text{Human} \sqcap \exists\text{hasChild}.\text{Parent}$
- $\text{Grandparent} \doteq \text{Human} \sqcap \exists\ \text{hasChild}.\exists\ \text{hasChild}.\top$
- **Uncle** $\doteq$ ....using Male, hasSibling, hasChild....

Roles

- $\mathtt{President} \doteq \mathtt{Person} \sqcap \exists\mathtt{presidentOf}.\top$
- $\mathtt{Independent} \doteq \mathtt{Person} \sqcap \neg\exists\mathtt{hasParty}.\top$
- $\mathtt{USPresident} \doteq \mathtt{Person} \sqcap \exists\mathtt{presidentOf}.\mathtt{UnitedStates}$
    - roles: presidentOf, hasParty…
    - universal concept ("top"): $\top$
    - existential restriction: $\exists$
- $\mathtt{Grandparent} \doteq \mathtt{Human} \sqcap \exists\mathtt{hasChild}.\mathtt{Parent}$
- $\mathtt{Grandparent} \doteq \mathtt{Human} \sqcap \exists\ \mathtt{hasChild}.\exists\ \mathtt{hasChild}.\top$
- $\mathtt{Uncle} \doteq \mathtt{Male} \sqcap \exists\ \mathtt{hasSibling}.\exists\ \mathtt{hasChild}.\top$

Null concept

- **Person ⊓ Group ⊑ ⊥**
  - null concept ("bottom"): ⊥
  - subsumption (sub concept): ⊑
- ⊑ is used for *subsumption axioms*
  - or: containment / specialisation axioms
- ≐ is used for *definitions* (or just ≡)
  - ≡ is also used for *equivalence axioms*
- Note the use of ... ⊑ ⊥ ("subsumption of bottom")
  to say that *something is not the case*

Null concept

- **Person** $\sqcap$ **Group** $\sqsubseteq \bot$
  - null concept ("bottom"): $\bot$
  - subsumption (sub concept): $\sqsubseteq$
- $\sqsubseteq$ is used for *subsumption axioms*
  - or: containment / specialisation axioms
- $\doteq$ is used for *definitions* (or just $\equiv$)
  - $\equiv$ is also used for *equivalence axioms*
- Note the use of ... $\sqsubseteq \bot$ ("subsumption of bottom") to say that something is not the case
- *This is a DL axiom*
  - so far we have just *defined* concepts

Axioms

- $\doteq$ is used for *definitions* of new concepts
- $\equiv$ is used for *equivalence axioms* about concepts
  - ...but some authors used it for *definitions* too :-/
- Axioms are equivalences or subsumptions:
  - *subsumption* axioms ($\sqsubseteq$):
    - composite concept (role) expressions on both sides
  - *equivalence axioms* ($\equiv$):
    - composite concept (role) expressions on both sides
    - corresponds to: $C \sqsubseteq D$, $D \sqsubseteq C$

More role definitions

- **LoyalRepublican** $\doteq$ **Republican** $\sqcap$ $\forall$**hasParty.Republican**
  - universal restriction: $\forall$
- **Monotheist** $\doteq$ **=1 believesInDeity.**$\top$
- **Polygamist** $\doteq$ **$\geq$3 hasSpouse.**$\top$
  - number restrictions: **=, $\geq$, $\leq$**
- **Narcissist** $\doteq$ $\exists$**hasLoveFor.<u>Self</u>**
  - **self references**: **<u>Self</u>**
- **MassMurderer** $\doteq$ ...using hasKilled, Human...

More uses of roles

- $\mathbf{LoyalRepublican} \doteq \mathbf{Republican} \sqcap \forall \mathbf{hasParty.Republican}$
  - universal restriction: $\forall$
- $\mathbf{Monotheist} \doteq\ =1\ \mathbf{believesInDeity}.\top$
- $\mathbf{Polygamist} \doteq \geq 3\ \mathbf{hasSpouse}.\top$
  - number restrictions: $=, \geq, \leq$
- $\mathbf{Narsissist} \doteq \exists \mathbf{hasLoveFor}.\underline{\mathbf{Self}}$
  - **self references**: $\underline{\mathbf{Self}}$
- $\mathbf{MassMurderer} \doteq \geq 4\ \mathbf{hasKilled.Human}$

Inverse and transitive roles

- $\texttt{StrayDog} \doteq \texttt{Dog} \sqcap \neg\exists\texttt{hasOwner}^{-}.\top$
- $\texttt{hasParent} \doteq \texttt{hasChild}^{-}$
- $\texttt{PureBred} \doteq \forall\texttt{hasParent}^{*}.\texttt{PureBred}$
  - inverse role: $\texttt{hasChild}^{-}$
  - transitive role: $\texttt{hasParent}^{*}$
- **Niece** $\doteq$ ..Woman, hasChild, hasSibling..

Inverse and transitive roles

- $\mathtt{StrayDog} \doteq \mathtt{Dog} \sqcap \neg\exists \mathtt{hasOwner}^{-}.\top$
- $\mathtt{hasParent} \doteq \mathtt{hasChild}^{-}$
- $\mathtt{PureBred} \doteq \forall \mathtt{hasParent}^{*}.\mathtt{PureBred}$
  - inverse role: $\mathtt{hasChild}^{-}$
  - transitive role: $\mathtt{hasParent}^{*}$
- $\mathtt{Niece} \doteq \mathtt{Woman} \sqcap \exists \mathtt{hasChild}^{-}.\mathtt{hasSibling}.\top$
- *We just define a role!*
  - until now, we have only defined *concepts*

Composite roles

- Similar to composite concepts, e.g.:
  - **holdsPresidency ≐ hasParty^{-} o presidentOf**
  - **hasLovedChild ≐ hasChild ⊓ hasLoveFor**
  - **hasBrother ≐ (hasSibling | Male)**
- Not all supported by OWL-DL and the reasoning engines
  - they can have “bad decision problems”
    - i.e., they compute slowly or intractably
  - ...with some exceptions
- **hasDaughter** ≐ ..using hasChild, Female..

Composite roles

- Similar to composite concepts, e.g.:
  - **holdsPresidency ≐ hasParty⁻ o presidentOf**
  - **hasLovedChild ≐ hasChild ⊓ hasLoveFor**
  - **hasBrother ≐ (hasSibling | Male)**
- Not all supported by OWL-DL and the reasoning engines
  - they can have “bad decision problems”
    - i.e., they compute slowly or intractably
  - ...with some exceptions
- **hasDaughter ≐ (hasChild | Female)**

Putting it together

- $\mathtt{Source} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{Content}$
- $\mathtt{TrustedContent} \doteq \exists\, \mathtt{hasSource}.\mathtt{TrustedSource}$
- $\mathtt{VerifiedContent} \doteq \exists\, \mathtt{verifiedBy}.\mathtt{FactChecker}$
- $\mathtt{DebunkedContent} \doteq \exists\, \mathtt{debunkedBy}.\mathtt{FactChecker}$
- $\mathtt{UnreliableSource} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{DebunkedContent}$
- $\mathtt{VerifyingSource} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{VerifiedContent} \sqcap \forall\, \mathtt{hasSource}^{-}.\mathtt{VerifiedContent}$

An *acyclic*, *definitional* TBox

Putting it together

- $\mathtt{Source} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{Content}$
- $\mathtt{TrustedContent} \doteq \exists\, \mathtt{hasSource}.\mathtt{TrustedSource}$
- $\mathtt{VerifiedContent} \doteq \exists\, \mathtt{verifiedBy}.\mathtt{FactChecker}$
- $\mathtt{DebunkedContent} \doteq \exists\, \mathtt{debunkedBy}.\mathtt{FactChecker}$
- $\mathtt{UnreliableSource} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{DebunkedContent}$
- $\mathtt{VerifyingSource} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{VerifiedContent}$
  $\sqcap\ \forall\, \mathtt{hasSource}^{-}.\mathtt{VerifiedContent}$

Concept expressions of atomic concepts

Defined concepts

Acyclic and unequivocal!

Expanded definitional TBox

- Acyclicity: no cyclic definitions in the TBox ("Terminology Box")
- Unequivocality: each named defined term is only used on the l.h.s. of a single definition
- Concept expansion:
  - every concept can be written as an expression of only atomic concepts
  - algorithm:
    - start with the expression that defines the concept
    - recursively replace all the defined concepts used in the expression with their definitions
    - halt when only atomic concepts remain

Expanded definitional TBox

- $\texttt{Source} \doteq \exists\, \texttt{hasSource}^{-}.\texttt{Content}$
- $\texttt{TrustedContent} \doteq \exists\, \texttt{hasSource}.\texttt{TrustedSource}$
- $\texttt{VerifiedContent} \doteq \exists\, \texttt{verifiedBy}.\texttt{FactChecker}$
- $\texttt{DebunkedContent} \doteq \exists\, \texttt{debunkedBy}.\texttt{FactChecker}$
- $\texttt{UnreliableSource} \doteq \exists\, \texttt{hasSource}^{-}.\exists\, \texttt{debunkedBy}.\texttt{FactChecker}$
- $\texttt{VerifyingSource} \doteq \exists\, \texttt{hasSource}^{-}.\exists\, \texttt{verifiedBy}.\texttt{FactChecker} \sqcap \forall\, \texttt{hasSource}^{-}.\exists\, \texttt{verifiedBy}.\texttt{FactChecker}$

Only basic concepts on the right hand sides!

Types of axioms

- Terminology axioms (TBox)
  - subsumptions: $C \sqsubseteq D$
  - equivalences: $C \equiv D$

    corresponds to: $C \sqsubseteq D,\ D \sqsubseteq C$
- Role axioms (RBox)
  - e.g., subsumptions: $R \sqsubseteq S$
- Individual assertion axioms (ABox)
  - class assertions: a:C
  - role assertions: <a,b>:R
- Knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ or $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$

C and D are *expressions*!

a and b are *individuals*.
R and S are *roles*!

Decision Problems

Reasoning over knowledge bases

- *What more can we do with ontologies?*
- For example:
  - given a *source ontology* that describes media content along with its sources and trustworthiness
  - we can *answer questions* like, e.g.:
    - is trusted content a type of content?
    - can content be both verified and debunked?
    - is all verified content trusted?
  - *competency questions* are what we want an ontology to answer
  - *DL offers a clear and compact way or representing and reasoning about questions such as these!*

Decision problems

- A computational problem with a yes/no answer, e.g.
  - is C *subsumed* by D? $\mathcal{K} \models C \sqsubseteq D$
  - are C and D *consistent*? $\mathcal{K} \models a:(C \sqcap D)$
  - are C and D *equivalent*? $\mathcal{K} \models C \equiv D$
  - are C and D *disjoint*? $\mathcal{K} \models C \sqcap D \sqsubseteq \bot$
  - does *a* *belong* to C: $\mathcal{K} \models a:C$?
  - is *a* *R-related* to *b:* $\mathcal{K} \models \langle a,b \rangle : R$?

C and D are classes,
a and b are individuals.
R is a role!

- Given a knowledge base $\mathcal{K}$, reasoning engines can give yes / no answers
  - ...but not all decision problems are *decidable*
  - ...or have tractable *complexity*
  - *depends on the expressions used!*

Decision problems in practice

- Description logic is implemented by *reasoning engines/OWL reasoner*
  - *e.g., HermiT and Pellet*
  - distinct from *inference engines*, such as OWL-RL
- Protegé-OWL
  - comes with HermiT, more plugins can be installed
- Solves decision problems, e.g.,
  - classifiy individuals
  - find subclass relationships (subsumptions)
  - find unsatisfiable classes (concepts)
  - detect inconsistencies

Manchester OWL syntax

Manchester OWL-syntax

- A simple DL notation without special symbols
  - used by Protege-OWL to construct classes
  - similar to DL syntax
- ```
  Class: InternalCensor
        EquivalentTo: Censor and Employee
  ```
- ```
  Class: ExternalCensor
        EquivalentTo: Censor and not Employee
  ```
- ```
  Class: Agent
        EquivalentTo: Person or Organisation or Group
  ```
- Can be used to *serialise* complete ontologies
  - ...we will look mostly at TBox expressions
- http://www.w3.org/TR/owl2-manchester-syntax/
  ```

Comparison

- DL:

```
ExternalCensor ≐ Censor ⊓ ¬Employee
```

- Machester OWL:

```
Class: ExternalCensor
    EquivalentTo: Censor and not Employee
```

- Turtle:

```
uib:ExternalCensor owl:equivalentClass
        owl:intersectionOf (
                uib:Censor
                [    a owl:Class ;
                     owl:complementOf uib:Employee
                ]
        ) .
```

Roles in Manchester OWL syntax

- `Class: President`
  `EquivalentTo:`
  `Person and presidentOf some owl:Thing`
- `Class: USPresident`
  `EquivalentTo:`
  `Person and presidentOf some UnitedStates`
- `Class: Independent`
  `EquivalentTo:`
  `Person and not hasParty some owl:Thing`
  - universal concept (top): `owl:Thing`
  - existential restriction: `some`

Null concept in Manchester OWL syntax

- **Class:** **<class-name>**

  EquivalentTo: Person and Group
  SubClassOf: owl:Nothing

  - null concept (bottom): **owl:Nothing**
  - subsumption (subconcept): **SubClassOf:**
  - equivalence: **EquivalentTo:**
    - ...used both for *definitions* and for *axioms*

More roles in Manchester OWL syntax

- ```
  Class: LoyalRepublican
      EquivalentTo:
      Person and hasParty only RepublicanParty
  ```
  - value restriction: only
- ```
  Class: Monotheist
      EquivalentTo: Person and
                    believesInDeity exactly 1
  ```
- ```
  Class: Polygamist
      EquivalentTo: hasSpouse min 3
  ```
  - number restriction: exactly, min, max
- ```
  Class: Narcissist
      EquivalentTo: loves some Self
  ```

Inverse, symmetric and transitive roles

- Class: StrayDog
  EquivalentTo: Dog and not inverse hasOwner some Person
- Class: hasParent
  EquivalentTo: inverse hasChild
- ObjectProperty: hasSibling
  Characteristics: Symmetric
- ObjectProperty: hasAncestor
  Characteristics: Transitive
- inverse role: inverse
- symmetric role:
  Characteristics: SymmetricProperty
- transitive role:
  Characteristics: TransitiveProperty

After Easter:
Graph Embeddings