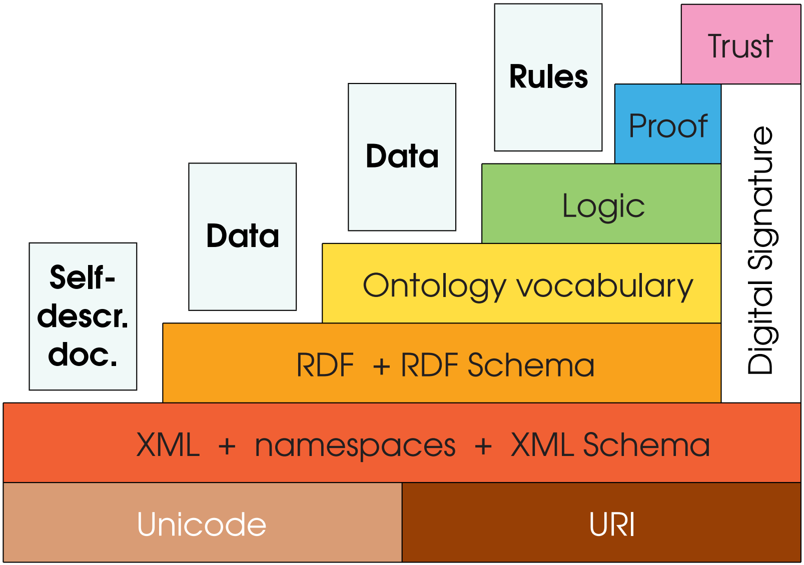
Layered approach to the Semantic Web

(by T. Berners-Lee)

semantics + reasoning

relational data

information exchange



In the context of Semantic Web, ontologies are expected to play an important role in helping automated processes (‘intelligent agents’) to access information. In particular, ontologies are expected to be used to provide structured vocabularies that explicate the relationships between different terms, allowing intelligent agents (and humans) to interpret their meaning flexibly yet unambiguously

Requirements for ontology languages

- a well-defined **syntax**

  it is a necessary condition for machine-processing of information

  all the languages we have considered so far have a well-defined syntax

- a **formal semantics** (describe the meaning of knowledge **precisely**)

  one use of a formal semantics is to allow people—and computers—to **reason** about the knowledge:

  - class membership: if $\underbrace{x \in C}_{x \text{ is an instance of } C}$ and $\underbrace{C \subseteq D}_{C \text{ is a subclass of } D}$, then we can infer $x \in D$
  - consistency: $x \in A$, $A \subseteq B \cap C$, $A \subseteq D$ and $\underbrace{B \cap D = \emptyset}_{B \text{ and } D \text{ are disjoint}}$,

    then **inconsistency!** (since A is empty)
  - classification: if certain property-values pairs are a sufficient condition for membership in A and x satisfies them, we can conclude that $x \in A$

- efficient **reasoning support** (derivations can be made mechanically)

  consistency, unintended relationships between classes, classification

- sufficient **expressive power** for applications

Limitations of the expressive power of RDFS

RDF/S allow the representation of **some** ontological knowledge

(organisation of vocabularies in typed hierarchies:
subclass/subproperty, domain/range restrictions, instances of classes)

However, a number of other features are **missing**:

- **local scope of properties**
  cows eat only *plants*, while other *animals* may eat *meat*, too
- **disjointness of classes**
  male and *female* are disjoint
- **Boolean combinations** of classes (union, intersection and complement)
  person is the disjoint union of *male* and *female*
- **cardinality restrictions**
  a person has exactly two parents
- special characterisations of properties:
  transitive, **functional**, or the **inverse** of another property
  `greater than', `has mother', `eats' and `is eaten by'

From RDF to OWL

- Two languages have been developed to satisfy the requirements:
  - **OIL** (Ontology Inference Layer):
    developed by a group of (largely) European researchers (several from the EU OntoKnowledge project) `http://www.ontoknowledge.org/oil/`
  - **DAML-ONT** (DARPA Agent Markup Language):
    developed by a group of (largely) US researchers (in DARPA DAML programme)
    `http://www.daml.org/`
- Efforts merged to produce **DAML+OIL**
  - development was carried out by
    'Joint EU/US Committee on Agent Markup Languages'
  - extends ('DL subset' of) RDF/S
- DAML+OIL was submitted to W3C as a basis for standardisation
  - Web-Ontology (WebOnt) Working Group formed
  - WebOnt group developed the **OWL** language based on DAML+OIL
  - OWL 1.0 language is a W3C Recommendation (since February 2004)
    `http://www.w3.org/TR/owl-features/`
- **OWL 2** is a **W3C Recommendation** since 2009
  `http://www.w3.org/TR/owl2-overview/`

What does the acronym 'OWL' stand for?

Actually, OWL is not a real acronym. The language started out as the 'Web Ontology Language' but the Working Group disliked the acronym 'WOL'

Owl lived at The Chestnuts, an old-world residence of great charm, which was grander than anybody else's, or seemed so to Bear, because it had both a knocker and a bell-pull. Underneath the knocker there was a notice which said:

PLES RING IF AN RNSER IS REQIRD.

Underneath the bell-pull there was a notice which said:

PLEZ CNOKE IF AN RNSR IS NOT REQID.

These notices had been written by Christopher Robin, who was the only one in the forest who could spell; for Owl, wise though he was in many ways, able to read and write and spell his own name **WOL**, yet somehow went all to pieces over delicate words like MEASLES and BUTTEREDTOAST.

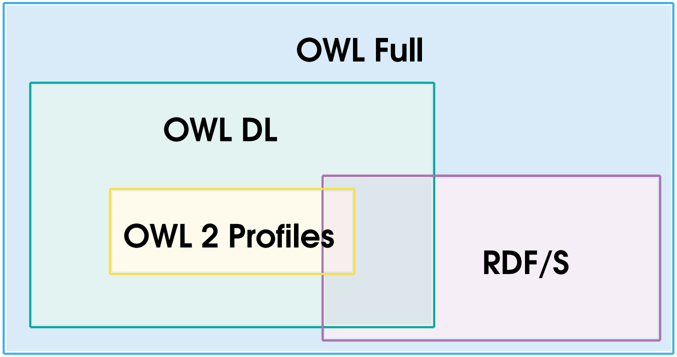
(*A.A. Milne, 'Winnie-the-Pooh'*)

Species of OWL

- **OWL Full**
  - uses all of the OWL language primitives
  - allows combinations of these primitives in arbitrary ways with RDF/S
    (including changing the meaning of the predefined (RDF or OWL) primitives, e.g., limiting the number of classes that can be described in any ontology)
  - is fully **upward-compatible** with RDF (syntactically and semantically):
    - any legal RDF document is also a legal OWL Full document
    - any valid RDF/S conclusion is also a valid OWL Full conclusion
  - is **undecidable** (no complete (or efficient) reasoning support)
- **OWL DL** (short for Description Logic)
  - is a sublanguage of OWL Full
    imposes restrictions on the use of OWL/RDF constructors:
    essentially, application of OWL's constructors to each other is disallowed
  - permits reasonably **efficient reasoning support**
- **OWL 2 Profiles**
  - sublanguages of OWL 2
    that trade expressive power for **efficiency of reasoning**
  - useful in different **application scenarios**
    OWL 2 EL (large ontologies), **OWL 2 RL** (rules), **OWL 2 QL** (ontology-based data access)

Species of OWL (cont.)

Upward compatibility between the three species of OWL and RDF/S:



- OWL uses RDF/S to a large extent:
  - all varieties of OWL use RDF for their syntax
  - instances are declared as in RDF (using RDF descriptions and typing information)
  - OWL constructors like **owl:Class**, **owl:DatatypeProperty** and **owl:ObjectProperty** are specialisations of their RDF/S counterparts

The OWL language

There are different syntactic forms of OWL:

- **RDF/XML** syntax see http://www.w3.org/TR/owl2-mapping-to-rdf/
  (used for interchange: can be written and read by all conformant OWL 2 software)
- **OWL/XML** syntax that does not follow the RDF conventions
  (more easily read by human users) see http://www.w3.org/TR/owl2-xml-serialization/
- **functional** syntax (used in the language specification document)
  (much more compact and readable) see http://www.w3.org/TR/owl2-syntax/
- graphic syntax based on the conventions of **UML**
  (Unified Modelling Language)
  (an easy way for people to become familiar with OWL)
- **Manchester** syntax
  (used in the Protégé editor) see http://www.w3.org/TR/owl2-manchester-syntax/
- **Description Logic** for OWL DL and the profiles

OWL ontologies: header

```
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xml:base="http://www.dcs.bbk.ac.uk/">

   <owl:Ontology rdf:about="">
      <rdfs:comment>An example OWL ontology</rdfs:comment>
      <owl:priorVersion rdf:resource="http://www.dcs.bbk.ac.uk/uni-old-ns"/>
      <owl:imports rdf:resource="http://www.dcs.bbk.ac.uk/person"/>
      <rdfs:label>DCSIS Ontology</rdfs:label>
   </owl:Ontology>

   ...

</rdf:RDF>
```

OWL namespace

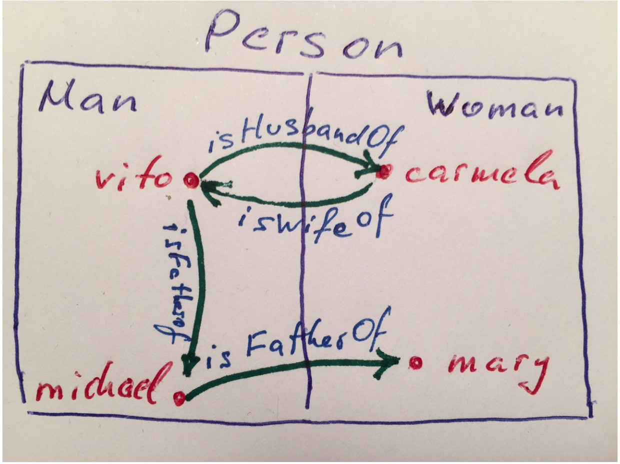
'Housekeeping' assertions

lists other ontologies whose content is assumed to be part of the current ontology

NB: while namespaces are used for disambiguation, imported ontologies provide definitions that can be used

Depicting individuals, classes, and properties

- **individuals** such as vito, carmela, michael, mary, etc.
- **classes** such as Man, Woman, Person, etc.
- **properties** such as isHusbandOf, isWifeOf, isFatherOf, etc.



Mathematical notation

$\text{vito} \in \text{Man}$

$\text{mary} \in \text{Woman}$

$(\text{vito}, \text{carmela}) \in \text{isHusbandOf}$

$\text{isHusbandOf} = \text{isWifeOf}^{-}$

$(\text{vito}, \text{michael}) \in \text{isFatherOf}$

$\text{Man} \subseteq \text{Person}$

$\text{Man} \cup \text{Woman} = \text{Person}$

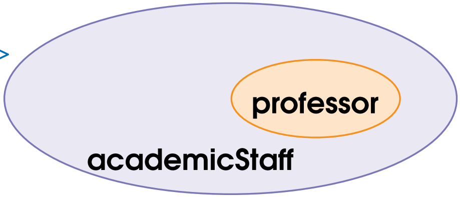
$\text{Man} \cap \text{Woman} = \emptyset$ (empty set)

The OWL language: classes

Classes are defined using an **owl:Class** element

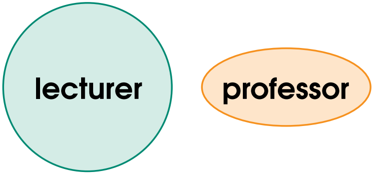
(**owl:Class** is a subclass of **rdfs:Class**)

```
<owl:Class rdf:ID="professor">
  <rdfs:subClassOf rdf:resource="#academicStaff"/>
</owl:Class>
```



$\text{professor} \sqsubseteq \text{academicStaff}$ Description Logic syntax

```
<owl:Class rdf:about="#professor">
  <owl:disjointWith rdf:resource="#lecturer"/>
</owl:Class>
```



$\text{professor} \sqcap \text{lecturer} \sqsubseteq \bot$

```
<owl:Class rdf:ID="faculty">
    <owl:equivalentClass rdf:resource="#academicStaff"/>
</owl:Class>
```

$\text{faculty} \equiv \text{academicStaff}$

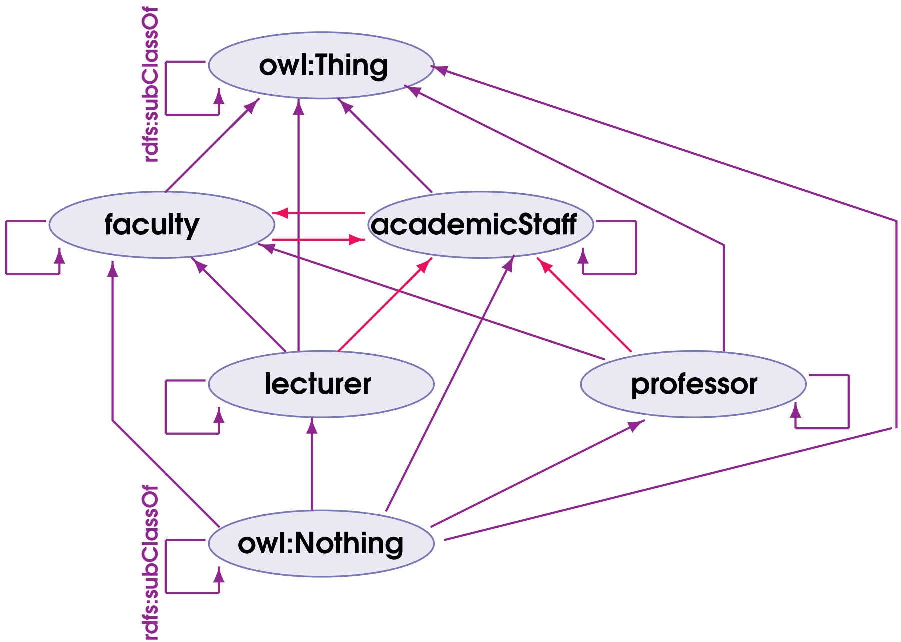
Q: is there another way to say that two classes are equivalent?

NB: there are two predefined classes: **owl:Thing** (contains everything) and **owl:Nothing** (the empty class)

The OWL language: classes

For every class C,

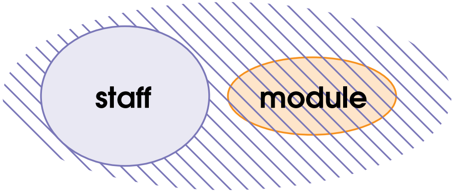
owl:Nothing is a subclass of C, and C is a subclass of **owl:Thing**



The OWL language: Boolean combinations

(i) “*modules* and *staff* members are disjoint”

```
<owl:Class rdf:about="#module">
  <owl:subClassOf>
    <owl:Class>
      <owl:complementOf rdf:resource="#staff"/>
    </owl:Class>
  </owl:subClassOf>
</owl:Class>
```



$\text{module} \sqsubseteq \neg\text{staff}$

Class: module SubClassOf: not staff

Manchester syntax

(ii) “*academic staff* are *lecturers*, *senior lecturers*, *readers* and *professors*”

```
<owl:Class rdf:ID="academicStaff">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#lecturer"/>
    <owl:Class rdf:about="#seniorLecturer"/>
    <owl:Class rdf:about="#reader"/>
    <owl:Class rdf:about="#professor"/>
  </owl:unionOf>
</owl:Class>
```

$\textbf{academicStaff} \equiv \textbf{lecturer} \sqcup \textbf{seniorLecturer} \sqcup \textbf{reader} \sqcup \textbf{professor}$

Class: academicStaff EquivalentTo: **lecturer** or **seniorLecturer** or **reader** or **professor**

NB: the new class **is equal** to the union (not **a subclass** as in (i))

The OWL language: Boolean combinations (cont.)

(iii) "*administrative staff* are those *staff members* that are neither *academic* nor *technical support staff*"

```
<owl:Class rdf:ID="adminStaff">
    <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#staffMember"/>
        <owl:Class>
            <owl:complementOf>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#academicStaff"/>
                        <owl:Class rdf:about="#techSupportStaff"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:complementOf>
        </owl:Class>
    </owl:intersectionOf>
</owl:Class>
```

$\text{adminStaff} \equiv \text{staffMember} \sqcap \neg(\text{academicStaff} \sqcup \text{techSupportStaff})$

Class: adminStaff EquivalentTo: **staffMember and not (academicStaff or techSupportStaff)**

NB: **owl:complementOf**, **owl:unionOf** and **owl:intersectionOf** are properties with domain **owl:Class** and range **rdf:List**

The OWL language: enumerations

```
<owl:Class rdf:about="#weekdays">
    <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Sunday"/>
        <owl:Thing rdf:about="#Monday"/>
        ...
        <owl:Thing rdf:about="#Saturday"/>
    </owl:oneOf>
</owl:Class>
```

$\text{weekdays} \equiv \{\text{Sunday}, \text{Monday}, \ldots, \text{Saturday}\}$

Class: weekdays EquivalentTo: { **Sunday, Monday, . . . , Saturday** }

The class extension of a class described with **owl:oneOf** contains **exactly** the enumerated **individuals**, no more, no less

NB: <owl:Thing *rdf:about*="..."/> refers to some individual

(remember: all individuals are by definition instances of **owl:Thing**)

The OWL language: instances

- Instances of classes are declared as in RDF:

```
<rdf:Description rdf:ID="MZ">
    <rdf:type rdf:resource="#academicStaff"/>
</rdf:Description>
```

or

```
<academicStaff rdf:ID="MZ"/>
```

MZ : academicStaff

- OWL **does not adopt** the **Unique Name Assumption**

(one individual may have different IDs)

So it must be explicitly asserted if certain IDs name different individuals:

```
<owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
        <lecturer rdf:about="#PTW"/>
        <lecturer rdf:about="#MZ"/>
        <lecturer rdf:about="#SM"/>
    </owl:distinctMembers>
</owl:AllDifferent>
```

- but:

```
<rdf:Description rdf:about="#William_Jefferson_Clinton">
    <rdf:sameAs rdf:resource="#BillClinton"/>
</rdf:Description>
```

Exercise: represent these statements in the DL syntax

The OWL language: properties

There are two kinds of properties:

- **object properties**, which relate objects to other objects (e.g., **isTaughtBy** and **supervises**)
- **data type properties**, which relate objects to datatype values (e.g., **phone** and **age**)

```
<owl:DatatypeProperty rdf:ID="age">
    <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
</owl:DatatypeProperty>
```

NB: OWL does not have any predefined data types

Instead, it allows one to use XML Schema data types

NB: "&xsd;nonNegativeInteger" is an abbreviation for "http://www.w3.org/2001/XMLSchema#nonNegativeInteger"

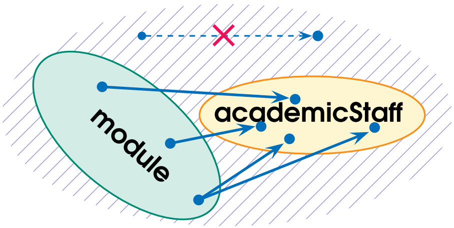
Such abbreviations can be defined using an ENTITY definition:

```
<!DOCTYPE rdf:RDF [ <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#"> ]>
```

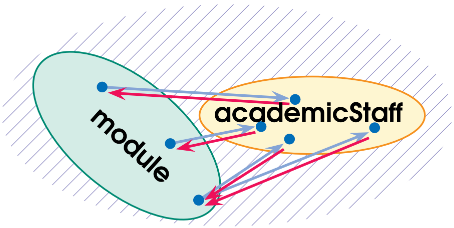
The OWL language: properties (cont.)

Object properties relate objects to other objects:

```
<owl:ObjectProperty rdf:about="#isTaughtBy">
    <rdfs:domain rdf:resource="#module"/>
    <rdfs:range rdf:resource="#academicStaff"/>
    <rdfs:subPropertyOf rdf:resource="#involves"/>
</owl:ObjectProperty>
```




```
<owl:ObjectProperty rdf:about="#teaches">
    <rdfs:domain rdf:resource="#academicStaff"/>
    <rdfs:range rdf:resource="#module"/>
    <owl:inverseOf rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
```




```
<owl:ObjectProperty rdf:about="#lecturesIn">
    <owl:equivalentProperty rdf:resource="#teaches"/>
</owl:ObjectProperty>
```

The OWL language: property restrictions

- “every *professor* must teach at least one *undergraduate module*”

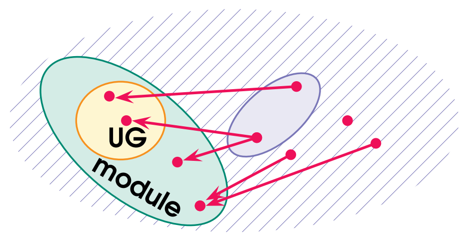
existential quantification

```
<owl:Class rdf:about="#professor">
    <owl:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#teaches"/>
            <owl:someValuesFrom rdf:resource="#undergraduateModule"/>
        </owl:Restriction>
    </owl:subClassOf>
</owl:Class>
```

$\text{professor} \sqsubseteq \exists \text{teaches.undergraduateModule}$

Class: professor SubClassOf: **teaches some undergraduateModule**

this **owl:Restriction** defines a class that consists of *all objects* for which **there exists** (at least one) **undergraduateModule** among values of **teaches**



NB: **owl:Restriction** defines an anonymous class which has no ID and has only local scope: it can only be used in the place where the restriction appears

The OWL language: property restrictions (cont.)

- "*teaching assistants* teach *postgraduate modules* only"

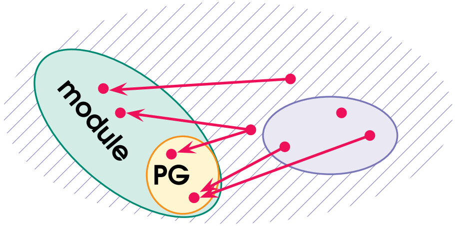
universal quantification

```
<owl:Class rdf:about="#teachingAssistant">
    <owl:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#teaches"/>
            <owl:allValuesFrom rdf:resource="#postgraduateModule"/>
        </owl:Restriction>
    </owl:subClassOf>
</owl:Class>
```

$\text{teachingAssistant} \sqsubseteq \forall \text{teaches.postgraduateModule}$

Class: teachingAssistant SubClassOf: **teaches only postgraduateModule**

this **owl:Restriction** defines a class that consists of *all objects* for which **all** values of teaches, if any, are from postgraduateModule



"if a teaching assistant teaches a module, then this module is a postgraduate module"

"a teaching assistant cannot teach a module that is not a postgraduate module"

$\text{teachingAssistant} \sqsubseteq \neg\exists \text{teaches}.\neg\text{postgraduateModule}$

http://en.wikipedia.org/wiki/Universal_quantification

The OWL language: property restrictions (cont.)

- “a *department* must have <u>at least 10</u> and <u>at most 30</u> academic staff members“

```
<owl:Class rdf:about="#department">
  <owl:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMember"/>
      <owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">
                                              10</owl:minQualifiedCardinality>
      <owl:onClass rdf:resource="#academicStaff"/>
    </owl:Restriction>
  </owl:subClassOf>
  <owl:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMember"/>
      <owl:maxQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">
                                              30</owl:maxQualifiedCardinality>
      <owl:onClass rdf:resource="#academicStaff"/>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>
```

Class: department SubClassOf: **hasMember min 10 academicStaff**

department $\sqsubseteq\ \geq 10$ hasMember.academicStaff

Class: department SubClassOf: **hasMember max 30 academicStaff**

NB: owl:qualifiedCardinality may be used when both owl:minQualifiedCardinality and owl:maxQualifiedCardinality have the same number

Class: department SubClassOf: **hasMember exactly 21 academicStaff**

The OWL language: property restrictions (cont.)

- “*mathematics modules* are taught by *Michael Zakharyaschev*”

```
<owl:Class rdf:about="#mathModule">
    <owl:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#isTaughtBy"/>
            <owl:hasValue rdf:resource="#MZ"/>
        </owl:Restriction>
    </owl:subClassOf>
</owl:Class>
```

at least one value of
isTaughtBy
is equal to
MZ (individual)

Class: mathModule SubClassOf: **isTaughtBy value MZ**

$\text{mathModule} \sqsubseteq \exists \text{isTaughtBy}.\{\text{MZ}\}$

Important binary relations

A binary relation R on a set A is called

- **reflexive** if xRx, for all $x \in A$
- **irreflexive** if **not** xRx, for all $x \in A$
- **symmetric** if xRy implies yRx, for all $x, y \in A$
- **asymmetric** if xRy imply **not** yRx, for all $x, y \in A$
- **transitive** if xRy and yRz imply xRz, for all $x, y, z \in A$.

Which of the following relations on the set $\mathbb{N}$ of natural numbers are reflexive, symmetric, etc.:

$<$, $=$, $\leq$, $\neq$, 'x divides y' ?

The **inverse** of R is the relation R^- such that xR^-y iff yRx, for all x, y

What is the relation between the 'important properties' of R and R^-?

OWL 1.0: special properties

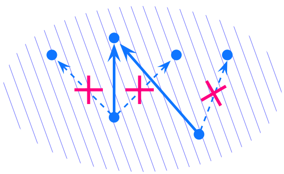
- **owl:TransitiveProperty** (e.g., isTallerThan, isAncestorOf)

  for all x, y, z, if $R(x,y)$ and $R(y,z)$ then $R(x,z)$

- **owl:SymmetricProperty** (e.g., isSiblingOf)

  for all x, y, if $R(x,y)$ then $R(y,x)$

R is functional



- **owl:FunctionalProperty**

  (at most one value for each object: e.g., directSupervisor)

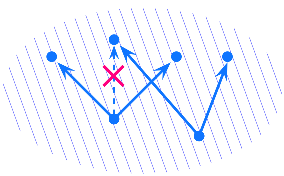
  for every x there is at most one y with $R(x,y)$

- **owl:InverseFunctionalProperty**

  (two different objects cannot have the same number: e.g., isTheSocialSecurtyNumberFor)

  for every y there is at most one x with $R(x,y)$

R is inverse functional



What's missing here?

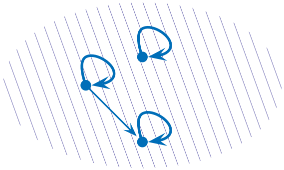
```
<owl:ObjectProperty rdf:ID="hasSameGradeAs">
    <rdfs:domain rdf:resource="#student"/>
    <rdfs:range rdf:resource="#student"/>
    <rdf:type rdf:resource="&owl;TransitiveProperty"/>
    <rdf:type rdf:resource="&owl;SymmetricProperty"/>
</owl:ObjectProperty>
```

OWL 2: more special properties

- **owl:ReflexiveProperty**

  (e.g., hasSameGrade)

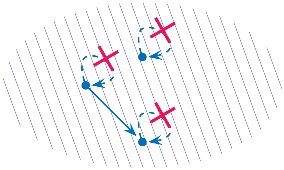
  for all x, $R(x, x)$



- **owl:IrreflexiveProperty**

  (e.g., isMotherOf)

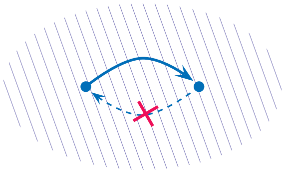
  for all x, not $R(x, x)$



- **owl:AsymmetricProperty**

  (e.g., isTallerThan)

  for every x, y, if $R(x, y)$ then not $R(y, x)$



OWL 2: more about properties

- **Property chains**

  if somebody owns an object, then they also own all parts of the object:

  for all x, y, z, if owns(x, y) and hasPart(y, z) then owns(x, z)

```
<owl:ObjectProperty rdf:ID="owns">
    <owl:propertyChainAxiom rdf:parseType="Collection">
        <owl:ObjectProperty rdf:about="#owns"/>
        <owl:ObjectProperty rdf:about="#hasPart"/>
    </owl:propertyChainAxiom>
</owl:ObjectProperty>
```

```
ObjectProperty: owns SubPropertyChain: owns o hasPart
```

- **Disjoint properties**

  for every x, y, either not isMotherOf(x, y) or not isSisterOf(x, y)

```
<owl:ObjectProperty rdf:about="#isMotherOf">
    <owl:propertyDisjointWith rdf:about="#isSisterOf"/>
</owl:ObjectProperty>
```

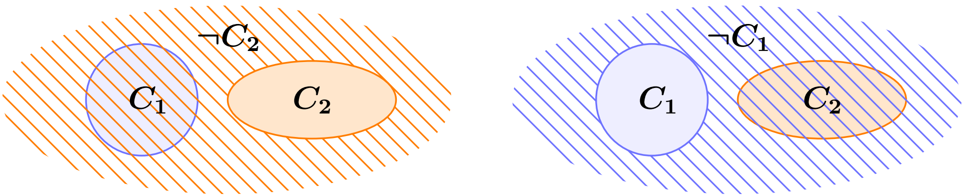
OWL (in functional syntax) as DL: Class Constructors

A	A
owl:Thing	$\top$ (top)
owl:Nothing	$\bot$ (bottom)
ObjectIntersectionOf$(C_1\ C_2\ \ldots C_n)$	$C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n$ (and)
ObjectUnionOf$(C_1\ C_2\ \ldots C_n)$	$C_1 \sqcup C_2 \sqcup \cdots \sqcup C_n$ (or)
ObjectComplementOf(C)	$\neg C$ (not)
ObjectOneOf$(a_1\ a_2\ \ldots a_n)$	$\{a_1\} \sqcup \{a_2\} \sqcup \cdots \sqcup \{a_n\}$
ObjectAllValuesFrom$(R\ \ C)$	$\forall R.C$ (all R-successors are in C)
ObjectSomeValuesFrom$(R\ \ C)$	$\exists R.C$ (an R-successor exists in C)
ObjectMinCardinality$(R\ \ n\ \ C)$	$\geq n\ R.C$ (there are at least n R-successors are in C)
ObjectMaxCardinality$(R\ \ n\ \ C)$	$\leq n\ R.C$ (there are at most n R-successors are in C)
ObjectHasValue$(R\ \ a)$	$\exists R.\{a\}$ (a is an R-successor)

OWL as DL: Classes

SubClassOf$(C \ \ D)$	$C \sqsubseteq D$ (all C are D)
EquivalentClasses$(C_1\ C_2 \dots C_n)$	$C_1 \equiv C_2, \quad C_2 \equiv C_3, \quad \dots, \quad C_{n-1} \equiv C_n$
DisjointClasses$(C_1\ C_2 \dots C_n)$	$C_1 \sqsubseteq \neg C_2,\ C_1 \sqsubseteq \neg C_3,\ \dots,\ C_1 \sqsubseteq \neg C_n$ $C_2 \sqsubseteq \neg C_3,\ \dots,\ C_2 \sqsubseteq \neg C_n$ $\dots$ $C_{n-1} \sqsubseteq \neg C_n$

Example: C_1 and C_2 are **disjoint**:

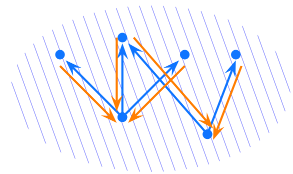


$C_1 \sqsubseteq \neg C_2$ is the same as $C_2 \sqsubseteq \neg C_1$

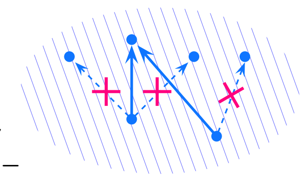
OWL as DL: Object Properties

SubObjectPropertyOf($R\ \ S$)	$R \sqsubseteq S$
EquivalentObjectProperties($R\ \ S$)	$R \equiv S$
InverseObjectProperties($R\ \ S$)	$R \equiv S^-$
TransitiveObjectProperty(R)	$R \circ R \sqsubseteq R$
FunctionalObjectProperty(R)	$\top \sqsubseteq\ \leq 1\ R$
InverseFunctionalObjectProperty(R)	$\top \sqsubseteq\ \leq 1\ R^-$
SymmetricObjectProperty(R)	$R^- \sqsubseteq R$
ObjectPropertyRange($R\ \ C$)	$\top \sqsubseteq \forall R.C$
ObjectPropertyDomain($R\ \ D$)	$\exists R.\top \sqsubseteq D$

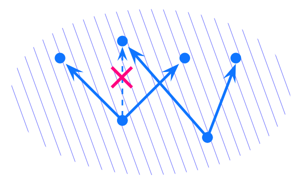
property R and its inverse R^-



R is functional



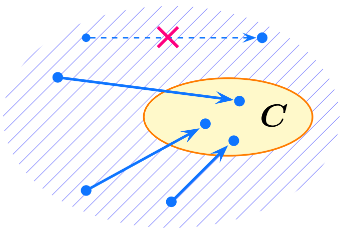
R is inverse functional



OWL as DL: Domain and Range Constraints

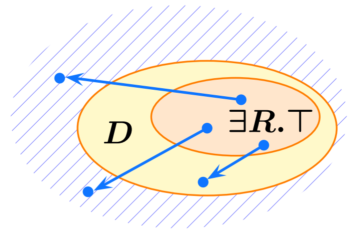
ObjectPropertyRange$(R\ \ C)$

$$\top \sqsubseteq \forall \boldsymbol{R}.\boldsymbol{C}$$



ObjectPropertyDomain$(R\ \ D)$

$$\exists \boldsymbol{R}.\top \sqsubseteq \boldsymbol{D}$$



NB: another way of representing the range constraint:

$$\exists \boldsymbol{R}^{-}.\top \sqsubseteq \boldsymbol{C}$$

the domain constraint:

$$\top \sqsubseteq \forall \boldsymbol{R}^{-}.\boldsymbol{D}$$

OWL as DL: Individuals

DifferentIndividuals$(a_1\ a_2\ \ldots a_n)$

$$\begin{aligned} & a_1 : \neg\{a_2, a_3, \ldots, a_n\} \\ & a_2 : \neg\{a_3, \ldots, a_n\} \\ & \ldots \\ & a_{n-1} : \neg\{a_n\} \end{aligned}$$

SameIndividuals$(a_1\ a_2\ \ldots a_n)$

$$\begin{aligned} & a_1 : \{a_2\} \\ & a_2 : \{a_3\} \\ & \ldots \\ & a_{n-1} : \{a_n\} \end{aligned}$$

The **Unique Name Assumption** (UNA) says that
any two individuals with different names are different individuals

- **an individual a is an instance of a class C** $\quad a : C$
- **an individual a is R-related to an individual b** $\quad (a, b) : R$
  R a property

OWL: summary

- OWL is the **proposed standard** for Web-ontologies. It allows us to describe the semantics of knowledge in a machine-accessible way

- OWL builds upon **RDF/S**: (XML-based) RDF syntax is used;
  instances are defined using RDF descriptions;
  and most RDF modelling primitives are used

- Formal **semantics** and **reasoning support** is provided through the mapping of OWL to logics.
  Predicate logic and description logics have been used for this purpose

Useful link: OWL tutorial at

`http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/`

Department ontology

(*classes* to be created)

1. "*first-year modules* are taught by *professors*"
2. "all *academic staff members* must teach at least one *undergraduate module*"
3. "a *department* must have <u>at least 10</u> and <u>at most 30</u> members"
4. "*mathematics modules* are taught by *MZ*"
5. "*modules* and *staff members* are disjoint"
6. "*personAtUni* are *staff members* and *students*"
7. "*administrative staff* are those *staff members* that are neither *academic* nor *technical support staff*"

Department ontology (cont.)

8. "*PhD students* are not allowed to teach *first-year modules*"
9. "MZ teaches SW and MfC"
10. "Each professor teaches exactly two modules"
11. "Academic staff members are PTW, MZ and SM"
12. "Professors can only teach MSc modules"
13. "Each academic staff member is either a lecturer, or a senior lecturer, or a reader, or a professor"
14. "Each module is taught by one person"
15. "Neither AP nor PTW teaches a maths module"
16. "Professors are ML, MZ, AP and PTW"