Assessed assignment

- Assessed assignment 1
  - Due 17 Feb.
  - 3 questions
  - Level 10 students answer Q1 and one other
- Q1 Understand an OWL ontology
  - Install Protégé and download the clothing.owl ontology from the KMM website
  - Answer parts i. to v. by editing (and saving) the ontology
    - » Submit the revised ontology electronically
  - Each part i. to v. also requires a written answer
  - Questions relate to a line of clothing inspired by the Beatles Sgt. Pepper album cover

Assessed assignment

- Q2 Describe an existing ontology
  - E.g. one covered in lecture 6
  - 500 words / 1 page (750 words for level 10)
  - Summarise concisely
    - » Include example concept definitions
- Q3 Describe the work of Linnaeus
  - 500 words / 1 page (750 words for level 10)

Manchester OWL Syntax

- Derived from the OWL abstract syntax, but less verbose
  - Aims to be easier to read and write
    - » Especially for non-logicians
  - Minimal use of ()
  - Allows DL expressions to be written in an English-like grammar, for email exchanges, GUIs etc
- Previously…
  - ALC / SHOIN / SHOIQ logical syntax: "∀R.C"
  - OWL abstract syntax: `< Restriction>`
    `< onProperty R> < allValuesFrom C >>`
- Manchester syntax: "R only C"

Manchester OWL Syntax

- Observed that DL syntax is cryptic
- Quantifier Role. Concept order can be confusing and misread:

Person ⊓ ∃eats.Meat

correct: Persons that eat (among other things) some Meat

incorrect: some Persons eat Meat

Manchester syntax is:

Person that eats some Meat [Person ⊓ ∃eats.Meat]

Person that eats only Meat [Person ⊓ ∀eats.Meat]

Manchester OWL Syntax

DL Syntax	Manchester Syntax	Example
$\neg C$	not C	not Male
$C \sqcup D$	C or D	Man or Woman
$C \sqcap D$	C and D	Parent and Man
$\forall R.C$	R only C	hasColleague only Professor
$\exists R.C$	R some C	hasColleague some Professor
$\geq n\ R$	R min n	hasColleague min 3
$\leq n\ R$	R max n	hasColleague max 3
$= n\ R$	R exactly 3	hasColleague exactly 3
$\exists R.\{a\}$	R value a	hasColleague value Fred

Manchester OWL Syntax

- In case of ambiguity of scope, a precedence order is defined (from highest to lowest):
  - some, all, value, min, max, exactly, that
  - not
  - and
  - or

> 'that' and 'and' are synonyms, but, syntactically, 'that' can be used only once, prior to a role expression: Class that role quantifier Class …

- Example:

  Person that
  hasChild some (Person and
  (hasChild only Man) and (hasChild some Person))

Manchester OWL Syntax

- 'onlysome' design pattern
  - Common to specify "eats some Meat and eats only Meat"
  - The onlysome pattern makes this easier to state

  E.g. Pizza that hasTopping onlysome [MozzarellaTopping, TomatoTopping]

  Is shorthand for

  Pizza that
  (hasTopping some MozzarellaTopping) and
  (hasTopping some TomatoTopping) and
  (hasTopping only (MozzarellaTopping or TomatoTopping))

Covering / Closure

OWL and Protégé

- Protégé 4 allows class expressions to be entered by typing the Manchester OWL
  - E.g. define VegetarianPizza

ALC

$$\text{Pizza} \sqcap \neg\, \exists \text{hasTopping.FishTopping} \sqcap \neg\, \exists \text{hasTopping.MeatTopping}$$

Manchester

```
Pizza
and not (hasTopping some FishTopping)
and not (hasTopping some MeatTopping)
```

Protégé 4 GUI

OWL 1.1 and 2

- The original OWL has been extended
  - OWL 1.1 and 2 are based on the SROIQ logic
  - Adds new ways to reason about roles R
  - Adds new cardinality constraints
  - http://www.w3.org/TR/2009/REC-owl2-overview-20091027/
- These extensions are seen as useful in applications and technically feasible
  - SRIOQ is decidable
- New:
  - Roles
  - Number restrictions
  - Proof
  - Syntax

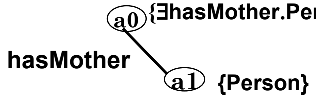
OWL: subPropertyOf

- subPropertyOf

  subPropertyOf(hasMother,hasParent)

  subPropertyOf(P, Q):: P(x, y) ⇒ Q(x, y)
- In DL: hasMother ⊑ hasParent

Show: ∃hasMother.Person ⊑ ∃hasParent.Person



Equivalently:
L(a0)= {∃hasMother.Person, ∀hasParent. ¬Person}
L(<a0,a1>)= {hasMother}
L(a1) = {Person}

By role inclusion: hasMother ⊑ hasParent
it should follow that:
L(<a0,a1>) = {hasMother, hasParent}
L(a1) = {Person, ¬Person}
Reasoning about roles increases expressivity

OWL 2: Roles

- Roles are given more prominence in OWL 2
- Role hierarchy
  - subPropertyOf
- Role assertions
  - Roles can be declared symmetric, transitive, reflexive or irreflexive
  - Disjoint roles, e.g. motherOf / sisterOf
- Role inclusion axioms
  - Propagate one property across another

  (owns ○ hasPart) ⊑ owns

  [i.e. owns(x, y) ∧hasPart(y, z) ⇒ owns(x, z)]

  Car ⊑ ∃hasPart.Engine *implies:*

  ∃owns.Car ⊑ ∃owns.Engine

OWL 2: Roles

- The set of roles is the set of role names, plus an inverse relation for each role name
- Formally, let **RN** be the set of role names

  the set of *roles* is **RN** ∪ {R^- | R ∈ **RN**}

  where R^- is the inverse role of R

$R^I \subseteq \Delta^I * \Delta^I$

$(R^-)^I = \{<y,x> \mid <x,y> \in R^I\}$

The function Inv() applies to roles:

Inv(R) = R^- Inv(R^-) = R

OWL 2: Roles

- The *Role box R* includes
  - The role hierarchy
  - Role inclusion axioms e.g. owns $\circ$ hasPart $\sqsubseteq$ owns
  - Role assertions
- Role inclusion axioms have some restrictions that prevent cyclic dependencies, these are valid:

  $R \circ S \sqsubseteq R$; $S \circ R \sqsubseteq R$; $R \circ R \sqsubseteq R$;

  $S^- \sqsubseteq S$

  More generally, $w \sqsubseteq R$ iff $Inv(w) \sqsubseteq Inv(R)$

  where w is a string of role names

OWL 2: Roles

- Role assertions

  Sym(R) if $\langle x,y\rangle \in R^I \Rightarrow \langle y,x\rangle \in R^I$

  Tra(R) if $(\langle x,y\rangle \in R^I$ and $\langle y,z\rangle \in R^I) \Rightarrow \langle x,z\rangle \in R^I$

  Ref(R) if $\{\langle x,x\rangle \mid x \in \Delta^I\} \subseteq R^I$

  Irr(R) if $R^I \cap \{\langle x,x\rangle \mid x \in \Delta^I\} = \varnothing$ *

  Dis(R,S) if $R^I \cap S^I = \varnothing$ *simple roles only
- In fact:

  Sym(R) = $R^- \sqsubseteq R$

  Tra(R) = $R \circ R \sqsubseteq R$

  So these role assertions are equivalent to inclusion axioms

Manchester OWL Syntax

- Manchester syntax has been extended to OWL 2
  - Property chains

    partOf $\circ$ partOf

    is written: partOf o partOf I.e. with an small letter o

OWL 2: Qualified number restrictions

- Number restrictions in OWL-DL
  - Minimum cardinality:

  $\geq n\ R$ :: $\{x \in \Delta^I \mid \#(\langle x,y\rangle \in R^I) \geq n\}$

  E.g. The set of things with at least 2 parts-that-are-Wheels:

  ≥ 2 hasWheel
- Number restrictions in OWL 2
  - Minimum cardinality specifies the class C for the n instances:

  $\geq n\ R.\ C$ :: $\{x \in \Delta^I \mid \#(\langle x,y\rangle \in R^I \wedge y \in C^I) \geq n\}$

  E.g. The set of things with at least 2 Wheels as parts:

  $\geq$ 2 hasPart. Wheel

  Similarly for maximum cardinality: $\leq n\ R.C$

  Simple roles only cannot say Tra(R) and $\geq n\ R.\ C$

OWL 2: Miscellaneous

- **Local reflexivity: ∃R.Self**
  - **e.g. ∃likes.Self**
  - **(∃R.Self)^I = {x| <x,x> ∈ R^I}**
- **Datatypes**
  - **dataOneOf {set} defines an enumerated datatype**
  - **dataComplementOf (data range) returns the complement of the data range**
  - **Datatype restriction uses datatype facet (from XML Schema)**
- **Annotations**
  - **Comments can be associated with subClassOf and axiom assertions**

OWL 2 Reasoning

- **As with ALC tableaux, goals are constructed and translated into negation normal form**
- **Additional equivalences:**
  - **¬(≤n R.C) = (≥(n+1) R.C)**
  - **¬(≥(n+1) R.C) = (≤n R.C)**
  - **¬(≥0 R.C) = ⊥**

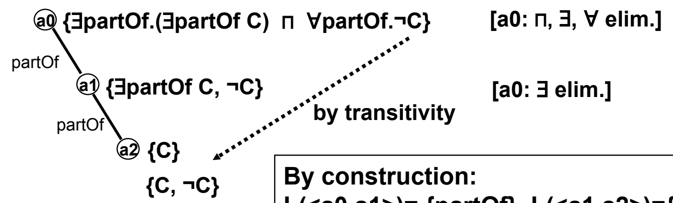
OWL 2 Reasoning

- **Reasoning about role hierarchies and axioms increases the number of tableaux rules**
  - **Recall there were only 4 rules for ALC, one for each operator**
  - **SROIQ has 18 rules**
  - **Automata theory is used to deal with role inclusion**
  - **Tableaux algorithm remains sound and complete**
    - **» Subsumption is reduced to unsatisfiability:**

      C ⊑ D iff C ⊓ ¬D ⊑ ⊥
  - **Blocking is used to terminate the algorithm**

OWL 2 Reasoning

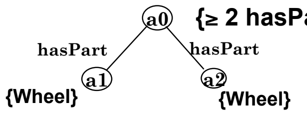
- **Reasoning about roles**



By construction:
L(<a0,a1>)= {partOf} L(<a1,a2>)={partOf}
By axiom (partOf ○ partOf) ⊑ partOf
∀partOf.¬C can be added to L(a1) and so
L(a2) = {C, ¬C} showing a contradiction

OWL 2 Reasoning

- **Number restrictions in OWL 2**
  - **The tableaux for SROIQ has generating rules and shrinking rules**
    - » **"The even more irresistible SROIQ" Horrocks, I., Kutz, O. and Sattler, U. KR 2006**



{≥ 2 hasPart.Wheel, ≤ 1 hasPart.Wheel }

1. Eliminate ≥2 to create 2 new nodes a1 a2
2. New rule for ≤nR.C Clash if > n nodes y, where L(y) includes C

OWL Functional Syntax

- **OWL 2 has a functional syntax and an XML syntax**
  - **XML syntax is not based on RDF/XML**
  - **XML schema is defined**
- **Functional Syntax Grammar**

Example 1: ∀**partOf**.**Car**
ObjectAllValuesFrom (**http://www.inf.org#partOf** **http://ww.inf.org#Car**)

Example 2: (owns ○ hasPart) ⊑ owns
SubObjectPropertyOf(
SubObjectPropertyChain(http://www.inf.org#owns http://www.inf.org#hasPart)
http://www.inf.org#owns)

OWL XML Syntax

Example 1: ∀**partOf**.**Car**
In XML:

```
<ObjectAllValuesFrom>
   <ObjectProperty IRI = "http://www.inf.org#partOf"/>
   <Class IRI = "http://www.inf.org#Car"/>
</ObjectAllValuesFrom>
```

Example 2:
: (owns ○ hasPart) ⊑ owns
In XML:

```
<SubObjectPropertyOf>
   <ObjectPropertyChain>
      <ObjectProperty IRI = "http://www.inf.org#owns"/>
      <ObjectProperty IRI = "http://www.inf.org#hasPart"/>
   </ObjectPropertyChain>
   <ObjectProperty IRI = "http://www.inf.org#owns"/>
</SubObjectPropertyOf>
```


OWL 2 Summary

- **OWL 2 extends OWL DL**
- **Adds the role box (hierarchy, assertions and inclusion axioms)**
- **Adds qualified number constraints**
- **Reasoning remains sounds and decidable**
- **XML syntax is based on a schema, not on RDF/XML**