UNIVERSITY OF Southampton
School of Electronics and Computer Science

Semantic Web in Depth: Web Ontology Language (OWL)

Dr Nicholas Gibbins
32/3019
nmg@ecs.soton.ac.uk

Introducing OWL

- For many, RDF Schema is a sufficiently expressive ontology language
- However, there are use cases which require a more expressive formalism:
  - Instance classification
  - Consistency checking
  - Subsumption reasoning

OWL Feature Summary

- Necessary and sufficient conditions for class membership
- Property restrictions
  - Local range, cardinality, value constraints
- Equivalence and identity relations
- Property characteristics
  - Transitive, symmetric, functional
- Complex classes
  - Set operators, enumerated classes, disjoint classes

OWL Versions

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Two versions of OWL:
  - OWL 1.0 (became Recommendation on 10 Feb 2004)
  - OWL 2 (became Recommendation on 29 Oct 2009)
- OWL 2 is more expressive than OWL 1.0, and takes advantage of developments in DL reasoning techniques in the intervening time
- We will initially concentrate on OWL 1.0

OWL 1.0 Species

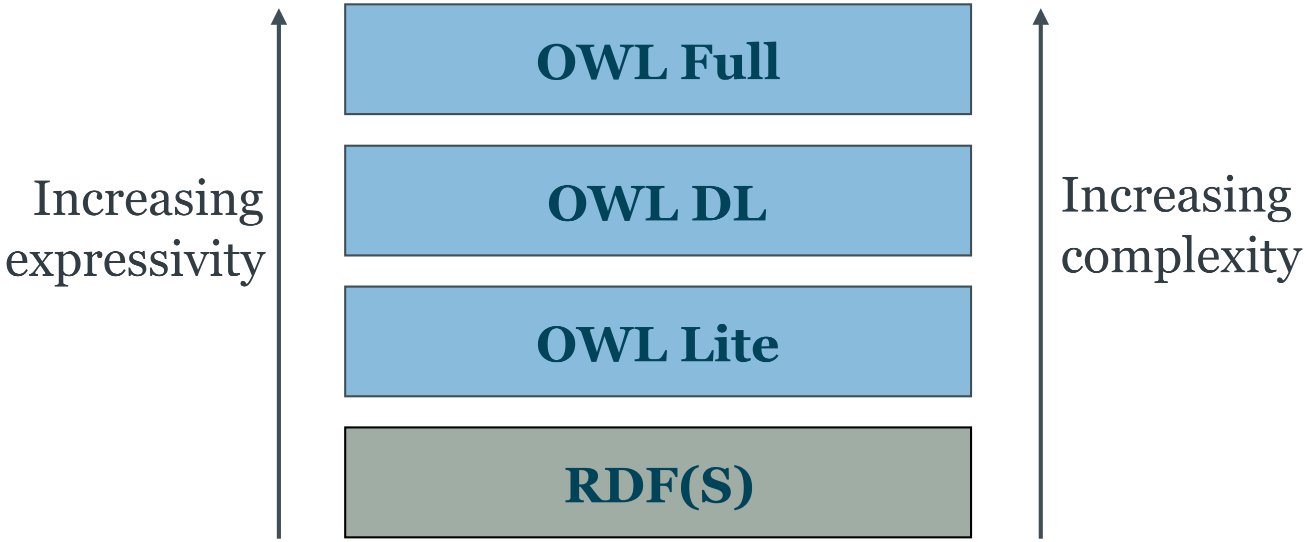
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Different subsets of OWL features give rise to the following sublanguages (colloquially known as **species**):
  - OWL Lite
  - OWL DL
  - OWL Full
- "There is a tradeoff between the expressiveness of a representation language and the difficulty of reasoning over the representations built using that language."

Brachman, R. J., and H. J. Levesque. (1984). The tractability of subsumption in frame-based description languages. In Proceedings of the 4th National Conference of the American Association for Artificial Intelligence (AAAI-84). Austin, TX, pp. 34-37.

OWL 1.0 Species

UNIVERSITY OF Southampton
School of Electronics and Computer Science



OWL Lite

- Description Logic-based
  - *SHIF*(D)
- Less complex reasoning at the expense of less expressive language
  - No enumerated classes, set operators, or disjoint classes
  - Restricted cardinality restrictions
    (values of 0 or 1 – required, permitted and excluded)
  - No value restrictions
  - equivalentClass/subClassOf cannot be applied to class expressions

OWL DL

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Description Logic-based
  - *SHOIN*(D)
  - Complete and decidable
  - Higher worst-case complexity than OWL Lite
- Supports all OWL constructs, with some restrictions
  - Properties that take datatype values cannot be marked as inverse functional
  - Classes, properties, individuals and datatype values are disjoint

OWL Full

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- No restrictions on use of language constructs
  - All OWL DL and RDFS constructs
- Potentially undecidable

University of Southampton
School of Electronics and Computer Science

OWL 1.0 Features and Syntax

Ontology header

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Ontology header for metadata

```
<owl:Ontology rdf:about="“”>
  <owl:versionInfo>1.4</owl:versionInfo>
  <rdfs:comment>An example ontology</rdfs:comment>
  <owl:imports
    rdf:resource="http://www.example.org/base"/>
</owl:Ontology>
```

Versioning support

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Version properties used in the ontology header
  - owl:versionInfo
    - Version number, etc
  - owl:priorVersion
    - Indicates that an ontology is a previous version of this
  - owl:backwardCompatibleWith
    - Indicates that the specified ontology is a previous version of this one, and that this is compatible with it
  - owl:incompatibleWith
    - Indicates that the specified ontology is a previous version of this one, but that this is incompatible with it

Versioning support

- Classes and properties may be marked as deprecated
  - owl:DeprecatedClass
  - owl:DeprecatedProperty

OWL class types

- owl:Class
  - Distinct from rdfs:Class – needed for OWL Lite/DL

- owl:Thing ($\top$)
  - The class that includes everything
- owl:Nothing ($\bot$)
  - The empty class

OWL property types

- owl:ObjectProperty
  - The class of resource-valued properties
- owl:DatatypeProperty
  - The class of literal-valued properties

- owl:AnnotationProperty
  - Used to type properties which annotate classes and properties (needed for OWL Lite/DL)

OWL versus RDF Schema

- Recall that the semantics of a description logic is specified by interpretation functions which map:
  - Instances to members of the domain of discourse
  - Classes to subsets of the domain of discourse
  - Properties to sets of pairs drawn from the domain of discourse
- Reflexive definitions of RDF Schema means that some resources are treated as both classes and instances, or instances and properties
- Ambiguous semantics for these resources
  - Can't tell from context whether they're instances or classes
  - Can't select the appropriate interpretation function
- The introduction of owl:Class, owl:ObjectProperty and owl:DatatypeProperty eliminates this ambiguity

OWL's Dirty Secret

UNIVERSITY OF
Southampton
School of Electronics
and Computer Science

OWL Full

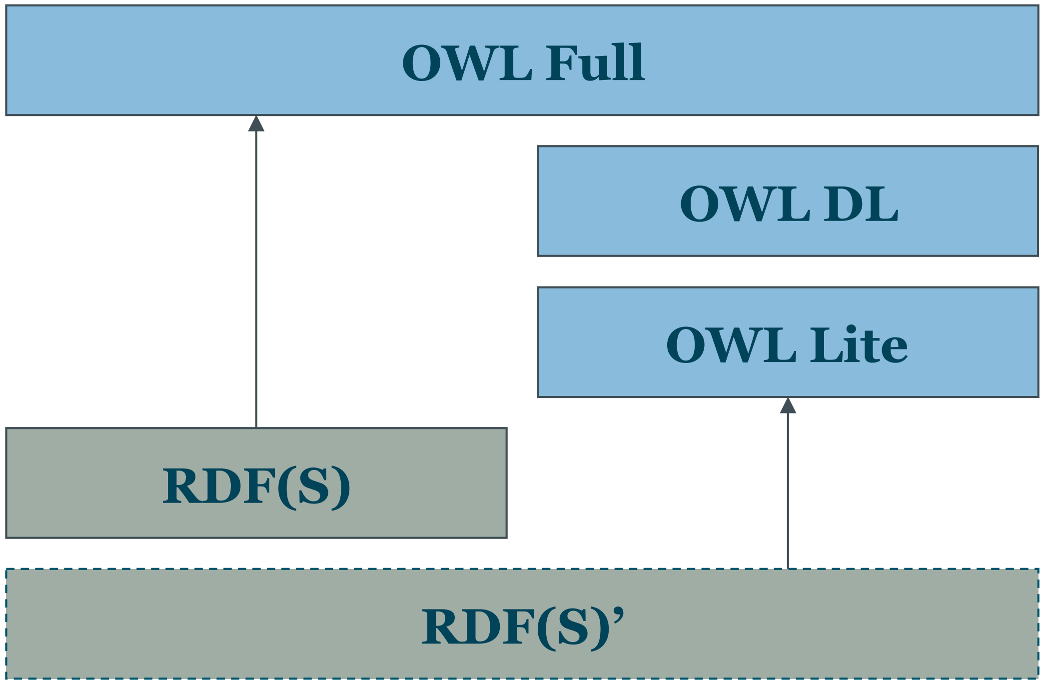
OWL DL

OWL Lite

RDF(S)

OWL's Dirty Secret Uncovered

UNIVERSITY OF Southampton
School of Electronics and Computer Science



OWL restrictions

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Class expression formed by constraints on properties
  - Local cardinality constraints
    $\leq n\ R$, $\geq n\ R$, $= n\ R$
  - Local range constraints
    $\exists R.C$, $\forall R.C$
  - Local value constraints
    $\exists R.\{x\}$
- Key concept in OWL

OWL restriction format

UNIVERSITY OF Southampton
School of Electronics
and Computer Science

```
<owl:Restriction>
  <owl:onProperty rdf:resource="property"/>
  constraint expression
</owl:Restriction>
```

Local cardinality constraints

- Defines a class based on the number of values taken by a property
- owl:minCardinality ($\geq$ n R)
  - "property R has at least n values"
- owl:maxCardinality ($\leq$ n R)
  - "property R has at most n values"
- owl:cardinality (= n R)
  - "property R has exactly n values"

- OWL Lite has restricted cardinalities

Local cardinality constraints

UNIVERSITY OF Southampton
School of Electronics
and Computer Science

- Single malt whiskies are whiskies which are distilled by one and only one thing

```
<owl:Class rdf:about="#SingleMaltWhisky">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Whisky"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#distilledBy"/>
          <owl:cardinality>1</owl:cardinality>
        </owl:Restriction>
      </owl:intersectionOf>
    <owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Local range constraints

- Defines a class based on the type of property values
- Distinct from global range constraint (rdfs:range) in RDF Schema
- owl:someValuesFrom ($\exists R.C$)
  - “there exists a value for property R of type C”
- owl:allValuesFrom ($\forall R.C$)
  - “property R has only values of type C”

- Can only be used with named classes or datatypes in OWL Lite

Local range constraints

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Carnivores are things which eat some things which are animals (∃eats.Animal)

```
<owl:Class rdf:about="#Carnivore">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eats"/>
      <owl:someValuesFrom rdf:resource="#Animal"/>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

Local range constraints

- Vegetarians are things which eat only things which are plants (∀eats.Plant)

```
<owl:Class rdf:about="#Vegetarian">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eats"/>
      <owl:allValuesFrom rdf:resource="#Plant"/>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

Local value constraints

- Defines a class based on the existence of a particular property value
- owl:hasValue ($\exists R.\{x\}$)
    - "property R has a value which is X"

- Cannot be used in OWL Lite

Local value constraints

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Green things are things which are coloured green
  ($\exists$ R. { Green })

```
<owl:Class rdf:about="#GreenThing">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasColour"/>
      <owl:hasValue rdf:resource="#Green"/>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

Set constructors

- owl:intersectionOf ($C \sqcap D$)
- owl:unionOf ($C \sqcup D$)
- owl:complementOf ($\neg C$)

- Restrictions on use with OWL Lite
  - owl:unionOf and owl:complementOf cannot be used
  - owl:intersectionOf can be used with named classes (not bNodes) and OWL restrictions only

Set constructors example

UNIVERSITY OF Southampton
School of Electronics and Computer Science

```
<owl:Class rdf:about="GreenApple">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="Apple">
        <owl:Restriction>
          <owl:onProperty rdf:resource="hasColor"/>
          <owl:hasValue rdf:resource="Green"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Equivalence and identity relations

- Useful for ontology mapping
  - owl:sameAs
  - owl:equivalentClass ($C \equiv D$)
  - owl:equivalentProperty ($R \equiv S$)

```
<owl:Thing rdf:about="#MorningStar">
  <owl:sameAs rdf:resource="#EveningStar"/>
</owl:Thing>
```

Non-equivalence relations

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- owl:differentFrom
  - Can be used to specify a limited unique name assumption

```
<rdf:Description rdf:about="#HarryCorbett">
  <owl:differentFrom rdf:resource="#HarryHCorbett"/>
</rdf:Description>
```

- OWL (and DLs in general) make the Open World Assumption
  - Knowledge of world is incomplete
  - If something cannot be proven true, then it isn't assumed to be false

Non-equivalence relations

UNIVERSITY OF Southampton
School of Electronics
and Computer Science

- owl:AllDifferent and owl:distinctMembers
  - Used to specify a group of mutually distinct individuals

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <rdf:Description rdf:about="#John"/>
    <rdf:Description rdf:about="#Paul"/>
    <rdf:Description rdf:about="#George"/>
    <rdf:Description rdf:about="#Ringo"/>
  </owl:distinctMembers>
</owl:AllDifferent>
```

Class Definitions

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Necessary Conditions ($\sqsubseteq$)
  - Primitive / partial classes
  - “If we know that something is a X, then it must fulfill the conditions...”
  - Defined using rdfs:subClassOf
- Necessary and Sufficient Conditions ($\equiv$)
  - Defined / complete classes
  - “If something fulfills the conditions..., then it is an X."
  - Defined using owl:equivalentClass

Property types - Inverse

- Defines a property as the inverse of another property ($R \equiv S^-$)

```
<owl:Property rdf:about="#hasAuthor">
  <owl:inverseOf rdf:resource="#wrote"/>
</owl:Property>
```

Property types - Symmetric

- Symmetric properties satisfy the axiom
  P(x,y) iff P(y,x)

```
<owl:SymmetricProperty rdf:about="#hasSibling"/>
```

Property types – Transitive

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Transitive properties satisfy the axiom
  P(x,y) and P(y,z) implies P(x,z)

```
<owl:TransitiveProperty rdf:about="#hasAncestor"/>
```

Property types – Functional

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Functional properties satisfy the axiom
  P(x,y) and P(x,z) implies y=z

```
<owl:FunctionalProperty rdf:about="#hasNINumber"/>
```

(everyone has only one NI number)

Property types – Inverse Functional

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Inverse functional properties satisfy the axiom
  P(y,x) and P(z,x) implies y=z

```
<owl:InverseFunctionalProperty rdf:about="#hasNINumber"/>
```

(people with the same NI number are the same person)

- Cannot be used with owl:DatatypeProperty in OWL Lite/DL

Disjoint classes

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- owl:disjointWith
  - members of one class cannot also be members of some specified other class

```
<owl:Class rdf:about="#MaleHuman">
  <rdfs:subClassOf rdf:resource="#Human"/>
  <owl:disjointWith rdf:resource="#FemaleHuman"/>
</owl:Class>
```

- Cannot be used in OWL Lite

Enumerated classes

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Defines a class as a direct enumeration of its members
  - owl:one of ($C \equiv \{a,b,c\}$)
- Cannot be extended (closed set)

- Cannot be used in OWL Lite

Enumerated classes example


```
<owl:Class rdf:about="#Continents">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Africa"/>
    <owl:Thing rdf:about="#Antarctica"/>
    <owl:Thing rdf:about="#Oceania"/>
    <owl:Thing rdf:about="#Europe"/>
    <owl:Thing rdf:about="#North-America"/>
    <owl:Thing rdf:about="#South-America"/>
    <owl:Thing rdf:about="#Asia"/>
  </owl:oneOf>
</owl:Class>
```

Ontology modularisation

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- owl:imports mechanism for including other ontologies
- Also possible to use terms from other ontologies without explicitly importing them
- Importing requires certain entailments, whereas simple use does not require (but also does not prevent) those entailments

Ontology modularisation example

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Ontology 1 (ont1) contains:
  BBB rdfs:subClassOf AAA
- Ontology-2 (ont2) contains:
  ont2 imports ont1
  CCC rdfs:subClassOf BBB

- Ontology-2 **must** entail
  CCC rdfs:subClassOf AAA

Ontology modularisation example

- Ontology 1 (ont1) contains:
  BBB rdfs:subClassOf AAA
- Ontology-3 (ont3) contains:
  CCC rdfs:subClassOf ont1:BBB

- Ontology-3 does **not necessarily** entail
  CCC rdfs:subClassOf ont1:AAA

OWL status

University of Southampton
School of Electronics and Computer Science

- WebOnt working group formed Nov 2001
- OWL Recommendations published in Feb 2004

OWL references

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Web Ontology Working Group homepage
  - http://www.w3.org/2001/sw/WebOnt/

UNIVERSITY OF Southampton
School of Electronics and Computer Science

OWL 2

From OWL 1 to OWL 2

UNIVERSITY OF Southampton
School of Electronics
and Computer Science

- OWL 1 design based on contemporary understanding of techniques for decidable, sound and complete reasoning in description logics
- Our understanding has improved since 2004
  - Some things that looked intractable have been shown to be possible
- Changes between 1 and 2 fall into the following categories:
  - Syntactic sugar (making it easier to say things we could already say)
  - Constructs for increased expressivity
  - Datatype support
  - Metamodelling
  - Annotation

Syntactic Sugar: Disjoint Union

- Allows us to define a class as the union of a number of other classes, all of which are pairwise disjoint

$$C \equiv C_1 \sqcup C_2 \sqcup \ldots \sqcup C_n$$

$$C1 \sqcap C2 \equiv \bot$$

- We'll look at this modelling pattern more in later lectures

Syntactic Sugar: Disjoint Classes

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- OWL 1 lets us state that two classes are disjoint
- OWL 2 lets us state that a set of classes are pairwise disjoint

Syntactic Sugar: Negative Property Assertions

- OWL 1 lets us assert property values for an individual
- OWL 2 lets us assert that an individual does not have a particular property value

```
<owl:NegativePropertyAssertion>
        <owl:sourceIndividual rdf:resource="john"/>
        <owl:assertionProperty rdf:resource="hasChild"/>
        <owl:targetIndividual rdf:resource="susan"/>
</owl:NegativePropertyAssertion>
```

New Constructs: Self Restriction

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Define a class of individuals which are related to to themselves by a given property

```
<owl:Restriction>
    <owl:onProperty rdf:resource="…"/>
    <owl:hasSelf rdf:datatype="&xsd;boolean">true</owl:hasSelf>
</owl:Restriction>
```

New Constructs: Qualified Cardinality Restrictions

- OWL 1 lets us either specify the local range of a property, or the number of values taken by the property
- OWL 2 lets us specify both together:

$$= 4\ hasPart.Wheel$$

```
<owl:Restriction>
        <owl:onProperty rdf:resource="hasPart"/>
        <owl:onClass rdf:resource="Wheel"/>
        <owl:cardinality rdf:datatype="&xsd;integer">4</owl:cardinality>
</owl:Restriction>
```

- Similar construct for datatype properties

New Constructs: Reflexive Properties

- Allows us to assert that a property is globally reflexive (relates every object to itself)

  <owl:ReflexiveProperty rdf:about="sameAgeAs"/>

New Constructs: Irreflexive Properties

- Allows us to assert that a property relates no object to itself

```
<owl:IrreflexiveProperty rdf:about="strictlyTallerThan"/>
```

New Constructs:
Asymmetric Properties

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Allows us to assert that a property is asymmetric:
  - If p(x,y), then not p(y,x)

```
<owl:AsymmetricProperty rdf:about="strictlyTallerThan"/>
```

New Constructs: Disjoint Properties

- Allows us to state that two individuals cannot be related to each other by two different properties that have been declared disjoint

```
<owl:ObjectProperty rdf:about="connectedTo">
        <owl:propertyDisjointWith rdf:resource="contiguousWith"/>
</owl:ObjectProperty>
```

New Constructs:
Property Chain Inclusion

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- OWL 1 does not let us define a property as a composition of other properties
  - Example: hasUncle ≡ hasParent o hasBrother

- OWL 2 lets us define such property compositions

```
<owl:ObjectProperty rdf:about="hasUncle">
    <owl:propertyChainAxiom rdf:parseType="Collection">
        <owl:ObjectProperty rdf:about="hasParent"/>
        <owl:ObjectProperty rdf:about="hasBrother"/>
    </owl:propertyChainAxiom>
</owl:ObjectProperty>
```

New Constructs: Keys

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- OWL 1 lets us define a property to be functional, so that individuals can be uniquely identified by values of that property
- OWL 2 lets us define uniquely identifying keys that comprise several properties

```
<owl:Class rdf:about="Person">
    <owl:hasKey rdf:parseType="Collection">
        <owl:DatatypeProperty rdf:about="hasSSN"/>
        <owl:DatatypeProperty rdf:about="birthDate"/>
    </owl:hasKey>
</owl:Class>
```

Datatype Restrictions

UNIVERSITY OF Southampton
School of Electronics
and Computer Science

- Allows us to define subsets of datatypes that constrain the range of values allowed by a datatype
- For example, the datatype of integers greater than or equal to 5:

```
<owl:Datatype>
    <owl:onDatatype rdf:resource="&xsd;integer"/>
    <owl:withRestrictions rdf:parseType="Collection">
        <xsd:minInclusive
            rdf:datatype="&xsd;integer">5</xsd:minInclusive>
    </owl:withRestrictions>
</owl:Datatype>
```

Metamodelling: Punning

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- OWL 1 required the names used to identify classes, properties, individuals and datatypes to be disjoint
- OWL 2 relaxes this
  - The same name (URI) can be used for both a class and an individual
- However:
  - A name cannot be used for both a class and a datatype
  - A name cannot be used for more than one type of property (DataProperty vs ObjectProperty)

Language Profiles

University of Southampton
School of Electronics and Computer Science

- OWL 1 has three dialects: OWL Lite, OWL DL and OWL Full

- OWL 2 introduces three profiles with useful computational properties (reasoning, conjunctive queries):
  - OWL 2 EL (PTIME-complete, PSPACE-complete)
  - OWL 2 QL (NLOGSPACE-complete, NP-complete)
  - OWL 2 RL (PTIME-complete, NP-complete)

  - OWL 1 DL (NEXPTIME-complete, decidability open)

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Manchester DL Syntax

A Plethora of Syntaxes

- The DL syntax we've used so far is a 'traditional' syntax for logical expressions

- Not well understood by non-logicians

- The Manchester DL syntax was introduced as a more user-friendly syntax for use in tools
  - Used in Protégé 4 – the subject of our next lecture

Manchester Syntax Summary

University of Southampton
School of Electronics and Computer Science

Traditional DL Syntax	Manchester Syntax
$C \sqcap D$	C and D
$C \sqcup D$	C or D
$\neg C$	not C
$\exists R.C$	R some C
$\forall R.C$	R only C
$\geq n\ R$	R min n
$\leq n\ R$	R max n
$= n\ R$	R exactly n
$\exists R.\{x\}$	R value x
$\geq n\ R.C$	R min n C
Reflexive property	R Self
Datatype restrictions	int[>=2, <=15]

UNIVERSITY OF Southampton

School of Electronics and Computer Science

The Protégé Ontology Editor

Protégé

- Leading ontology editor
- Early implementer of OWL (but was around before OWL)
- Thriving user community
    - Annual user conference
- Free and open source
    - http://protege.stanford.edu/
    - Many add-ons for visualisation, etc

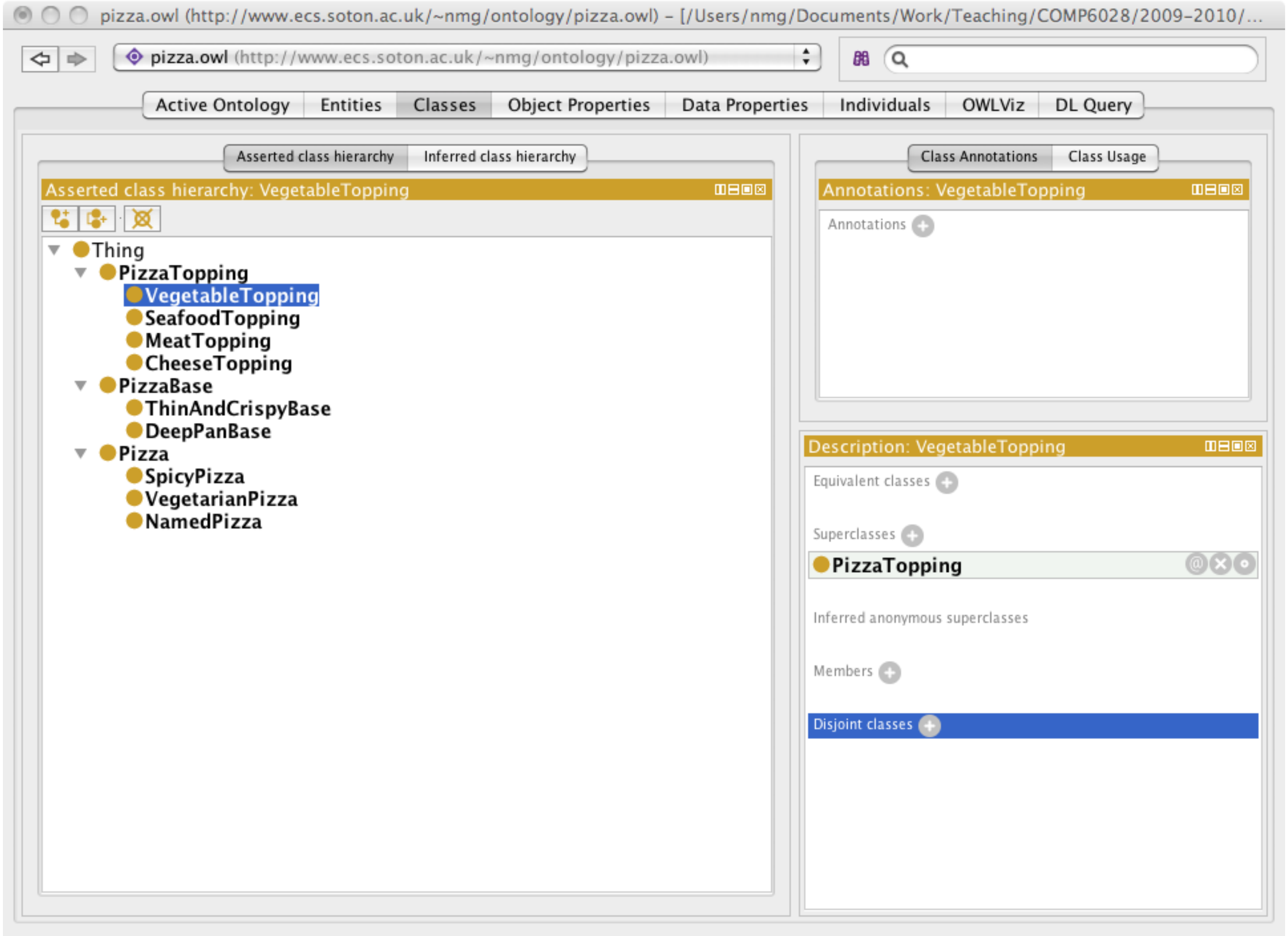
Protégé and DL Reasoners

- Protégé integrates reasoning into the ontology design process
  - Checks your ontology for consistency, subsumption, etc
  - Uses DIG interface to communicate with the reasoner
- Pellet
  - http://pellet.owldl.com/
- FaCT++
  - http://owl.man.ac.uk/factplusplus/

ESSENTIAL READING!

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Horridge et al, A Practical Guide to Building OWL Ontologies using the Protégé-OWL Plugin and CO-ODE Tools, 2007
- (available from COMP6028 website)


pizza.owl (http://www.ecs.soton.ac.uk/~nmg/ontology/pizza.owl) – [/Users/nmg/Documents/Work/Teaching/COMP6028/2009-2010/...
pizza.owl (http://www.ecs.soton.ac.uk/~nmg/ontology/pizza.owl)
Active Ontology
Entities
Classes
Object Properties
Data Properties
Individuals
OWLViz
DL Query
Asserted class hierarchy
Inferred class hierarchy
Asserted class hierarchy: VegetableTopping
Thing
PizzaTopping
VegetableTopping
SeafoodTopping
MeatTopping
CheeseTopping
PizzaBase
ThinAndCrispyBase
DeepPanBase
Pizza
SpicyPizza
VegetarianPizza
NamedPizza
Class Annotations
Class Usage
Annotations: VegetableTopping
Annotations
Description: VegetableTopping
Equivalent classes
Superclasses
PizzaTopping
Inferred anonymous superclasses
Members
Disjoint classes

Example ontology: OWL Pizzas

- Build an ontology for describing pizzas and their ingredients
- Must be able to determine whether pizzas are vegetarian, spicy, etc