Lecture Notes
Big Data in Medical Informatics

Week 3:
Ontologies

Oya Beyan, Ph.D.
Prof. Dr. Stefan Decker

RWTH AACHEN UNIVERSITY

Overview of OWL

What is an Ontology?

- formal specification of a certain domain
- machine manipulable model
- Ontology is about the exact description of things and their relationships and an inference mechanism for it.
- For the web, ontology is about
  - the exact description of web information and
  - relationships between web information and
  - reasoning with it.

- dictionary $\subset$ taxonomy $\subset$ ontology

OWL

- Web Ontology Language
- Based on predecessors (DAML+OIL)
- A Web Language: Based on RDF(S)
- An Ontology Language: Based on logic

- Three varieties in OWL 1.0 (Feb 2004)
  - OWL-full
  - OWL-DL (“OWL”)
  - OWL-Lite

- Three varieties in OWL 2.0 (Oct 2009)
  - EL – basis in the EL family of description logics
  - QL – allows reasoning by rewriting queries into a standard relational query language
  - RL – allows reasoning to be implemented using rule-based technologies

Why OWL?

- OWL is a part of the "Semantic Web Vision" - a future where:
  - Web information has exact meaning
  - Web information can be processed by computers
  - Computers can integrate information from the web

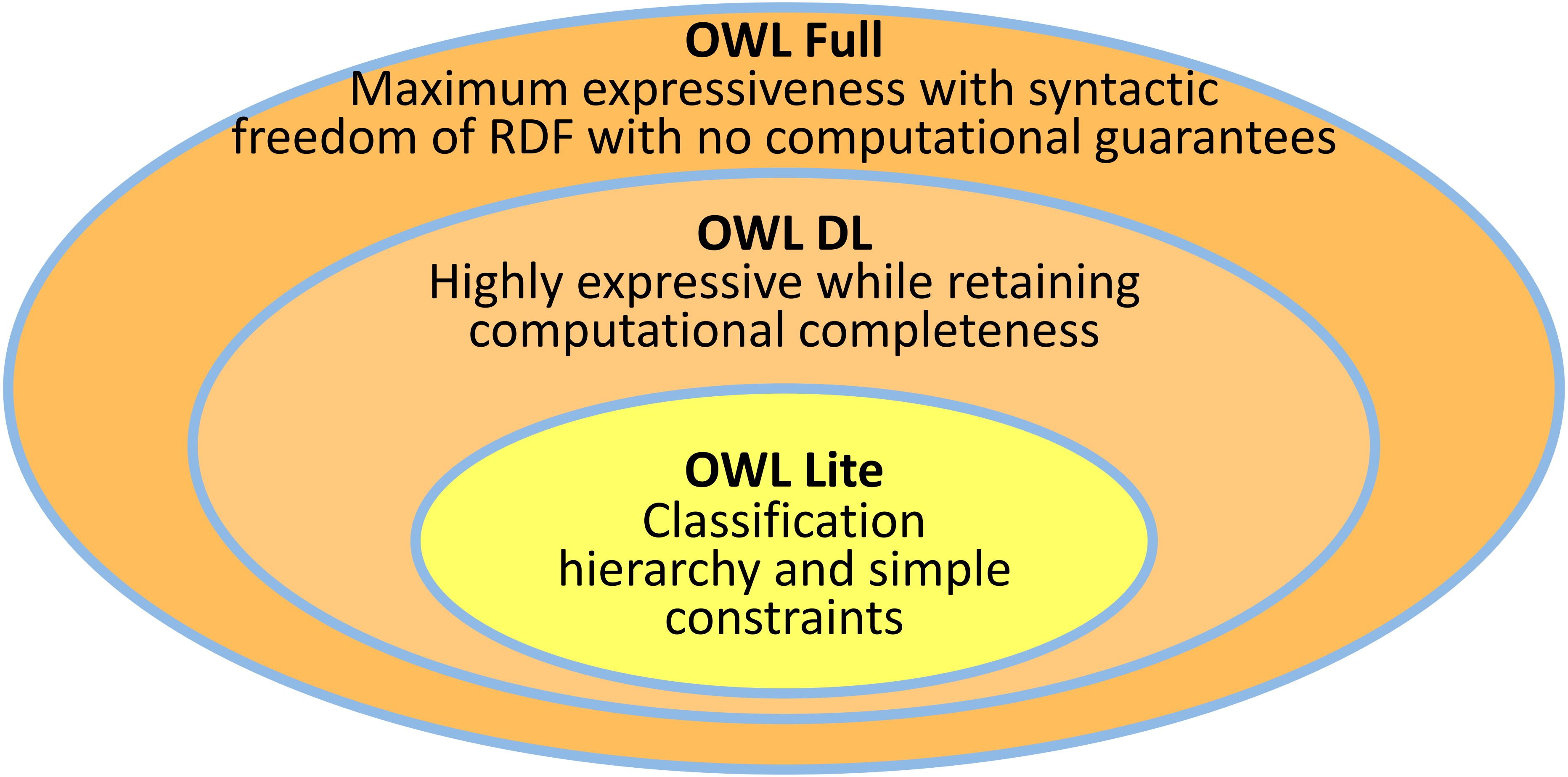
- OWL was designed to
  - provide a common way to process the content of web information (instead of displaying it).
  - be read by computer applications (instead of humans).

OWL Use

- At least two different user groups
  - OWL used as data exchange language

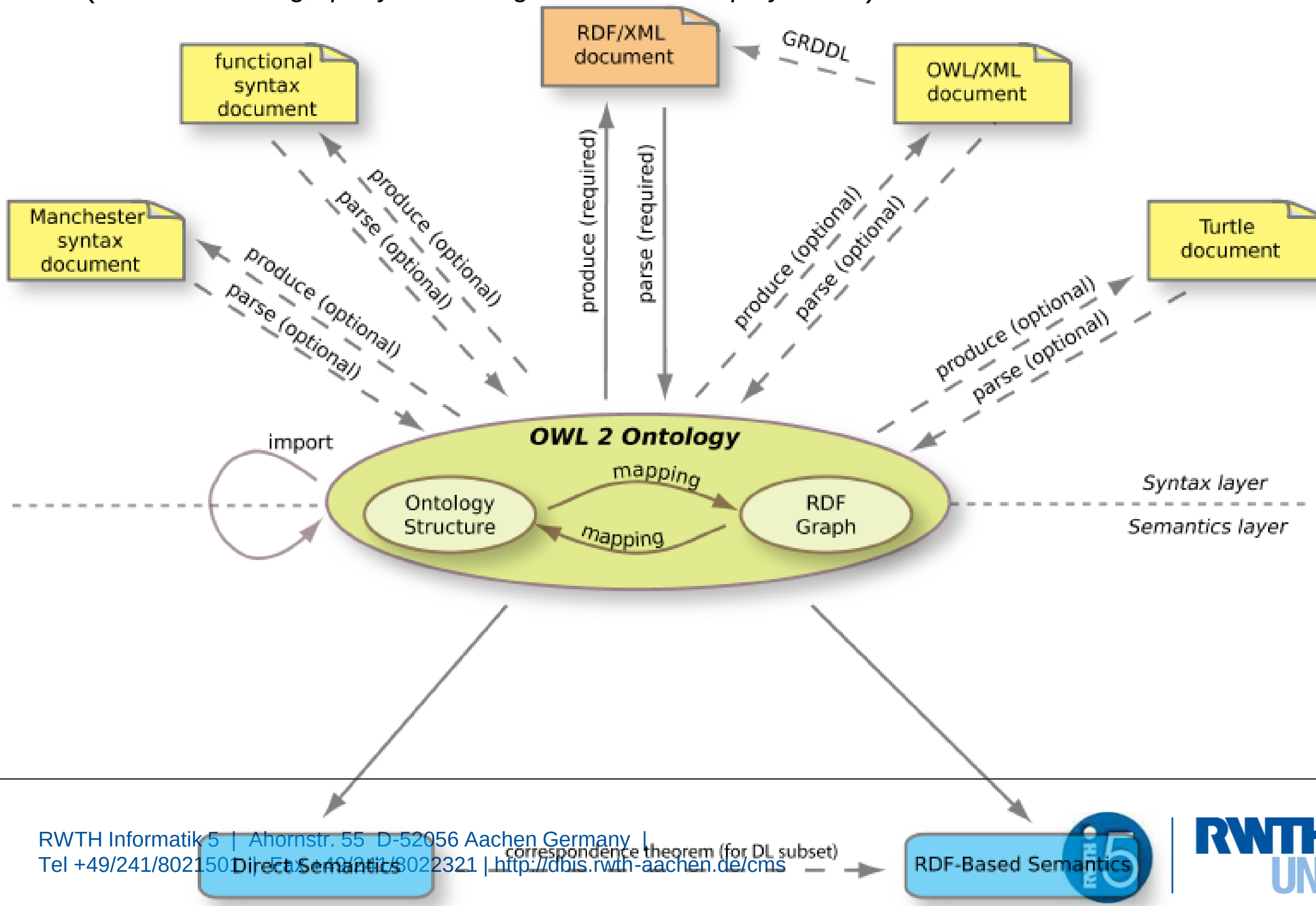
    (define interfaces of services and agents)
  - OWL used for terminologies or knowledge models

The Three Sublanguages of OWL 1.0



The three flavors of OWL 2.0

- OWL 2 EL (sound and complete reasoning in polynomial time)
- OWL 2 QL (Reasoning, query answering in Nlog time)
- OWL 2 RL (sound reasoning, query answering are worst-case polynomial)



OWL Constructs

W3C Recommendation

DisjointClasses(d_1 … d_n)	$EC(d_i) \cap EC(d_j) = \{\}$ for $1 \le i < j \le n$
EquivalentClasses(d_1 … d_n)	$EC(d_i) = EC(d_j)$ for $1 \le i < j \le n$
SubClassOf(d_1 d_2)	$EC(d_1) \subseteq EC(d_2)$
DatatypeProperty(p [*Deprecated*] *annotation*(p_1 o_1) … *annotation*(p_k o_k) *super*(s_1) … *super*(s_n) *domain*(d_1) … *domain*(d_n) *range*(r_1) … *range*(r_n) [*Functional*])	[$\langle S(c), S(\mathit{owl{:}DeprecatedProperty})\rangle \in ER(\mathit{rdf{:}type})$] $S(p) \in EC(\mathit{annotation}(p_1\ o_1))$ … $S(p) \in EC(\mathit{annotation}(p_k\ o_k))$ $ER(p) \subseteq O \times LV \cap ER(s_1) \cap \ldots \cap ER(s_n) \cap$ $EC(d_1) \times LV \cap \ldots \cap EC(d_n) \times LV \cap O \times EC(r_1) \cap \ldots \cap O \times EC(r_n)$ [ER(p) is functional]
ObjectProperty(p [*Deprecated*] *annotation*(p_1 o_1) … *annotation*(p_k o_k) *super*(s_1) … *super*(s_n) *domain*(d_1) … *domain*(d_n) *range*(r_1) … *range*(r_n) [*inverse*(i)] [*Symmetric*] [*Functional*] [*InverseFunctional*] [*Transitive*])	[$\langle S(c), S(\mathit{owl{:}DeprecatedProperty})\rangle \in ER(\mathit{rdf{:}type})$] $S(p) \in EC(\mathit{annotation}(p_1\ o_1))$ … $S(p) \in EC(\mathit{annotation}(p_k\ o_k))$ $ER(p) \subseteq O \times O \cap ER(s_1) \cap \ldots \cap ER(s_n) \cap$ $EC(d_1) \times O \cap \ldots \cap EC(d_n) \times O \cap O \times EC(r_1) \cap \ldots \cap O \times EC(r_n)$ [ER(p) is the inverse of ER(i)] [ER(p) is symmetric] [ER(p) is functional] [ER(p) is inverse functional] [ER(p) is transitive]
AnnotationProperty(p *annotation*(p_1 o_1) … *annotation*(p_k o_k))	$S(p) \in EC(\mathit{annotation}(p_1\ o_1))$ … $S(p) \in EC(\mathit{annotation}(p_k\ o_k))$
OntologyProperty(p *annotation*(p_1 o_1) … *annotation*(p_k o_k))	$S(p) \in EC(\mathit{annotation}(p_1\ o_1))$ … $S(p) \in EC(\mathit{annotation}(p_k\ o_k))$
EquivalentProperties(p_1 … p_n)	$ER(p_i) = ER(p_j)$ for $1 \le i < j \le n$
SubPropertyOf(p_1 p_2)	$ER(p_1) \subseteq ER(p_2)$
SameIndividual(i_1 … i_n)	$S(i_j) = S(i_k)$ for $1 \le j < k \le n$
DifferentIndividuals(i_1 … i_n)	$S(i_j) \ne S(i_k)$ for $1 \le j < k \le n$
Individual([i] *annotation*(p_1 o_1) … *annotation*(p_k o_k) *type*(c_1) … *type*(c_m) pv_1 … pv_n)	EC(*Individual*([i] *annotation*(p_1 o_1) … *annotation*(p_k o_k) *type*(c_1) … *type*(c_m) pv_1 … pv_n)) is nonempty

OWL Guide	OWL Reference	OWL Semantics
owl:AllDifferent / 4.3.	owl:AllDifferent	owl:AllDifferent
owl:allValuesFrom / 3.4.1.	owl:allValuesFrom	owl:allValuesFrom
owl:AnnotationProperty / 2.2.	owl:AnnotationProperty	owl:AnnotationProperty
owl:backwardCompatibleWith / 6.	owl:backwardCompatibleWith	owl:backwardCompatibleWith
owl:cardinality / 3.4.2.	owl:cardinality	owl:cardinality
owl:Class / 3.1.1.	owl:Class	owl:Class
owl:complementOf / 5.1.3.	owl:complementOf	owl:complementOf
	owl:DataRange	owl:DataRange
owl:DatatypeProperty / 3.2.2.	owl:DatatypeProperty	owl:DatatypeProperty
owl:DeprecatedClass / 6.	owl:DeprecatedClass	
owl:DeprecatedProperty / 6.	owl:DeprecatedProperty	
owl:differentFrom / 4.3.	owl:differentFrom	owl:differentFrom
owl:disjointWith / 5.3.	owl:disjointWith	owl:disjointWith
owl:distinctMembers / 4.3.	owl:distinctMembers	owl:distinctMembers
owl:equivalentClass / 4.1.	owl:equivalentClass	owl:equivalentClass
owl:equivalentProperty / 4.1.	owl:equivalentProperty	owl:equivalentProperty
owl:FunctionalProperty / 3.3.	owl:FunctionalProperty	owl:FunctionalProperty
owl:hasValue / 3.4.3.	owl:hasValue	owl:hasValue
owl:imports / 2.2.	owl:imports	owl:imports
owl:incompatibleWith / 6.	owl:incompatibleWith	owl:incompatibleWith
owl:intersectionOf / 5.1.1.	owl:intersectionOf	owl:intersectionOf
owl:InverseFunctionalProperty / 3.3.	owl:InverseFunctionalProperty	owl:InverseFunctionalProperty
owl:inverseOf / 3.3.	owl:inverseOf	owl:inverseOf
owl:maxCardinality / 3.4.2.	owl:maxCardinality	owl:maxCardinality
owl:minCardinality / 3.4.2.	owl:minCardinality	owl:minCardinality
owl:Nothing / 3.1.1.	owl:Nothing	owl:Nothing
owl:ObjectProperty / 3.2.1.	owl:ObjectProperty	owl:ObjectProperty
owl:oneOf / 5.2.	owl:oneOf	owl:oneOf
owl:onProperty / 3.4.	owl:onProperty	owl:onProperty
owl:Ontology / 2.2.	owl:Ontology	owl:Ontology
	owl:OntologyProperty	owl:OntologyProperty
owl:priorVersion / 6.	owl:priorVersion	owl:priorVersion
owl:Restriction / 3.4.	owl:Restriction	owl:Restriction
owl:sameAs / 4.2.	owl:sameAs	owl:sameAs
owl:someValuesFrom / 3.4.1.	owl:someValuesFrom	owl:someValuesFrom
owl:SymmetricProperty / 3.3.	owl:SymmetricProperty	owl:SymmetricProperty
owl:Thing / 3.1.1.	owl:Thing	owl:Thing
owl:TransitiveProperty / 3.3.	owl:TransitiveProperty	owl:TransitiveProperty
owl:unionOf / 5.1.2.	owl:unionOf	owl:unionOf
owl:versionInfo / 6.	owl:versionInfo	owl:versionInfo
		rdf:List
		rdf:nil
rdf:type		rdf:type
rdfs:comment / 2.2.		rdfs:comment
rdfs:Datatype / 3.2.2.		rdfs:Datatype
rdfs:domain / 3.2.1.	rdfs:domain	rdfs:domain
rdfs:label / 3.1.1.		rdfs:label
rdfs:Literal / 3.3.		rdfs:Literal
rdfs:range / 3.2.1.	rdfs:range	rdfs:range
rdfs:subClassOf / 3.1.1.	rdfs:subClassOf	rdfs:subClassOf
rdfs:subPropertyOf / 3.2.1.	rdfs:subPropertyOf	rdfs:subPropertyOf

RWTH Informatik 5 | Ahornstr. 55 D-52056 Aachen Germany |
Tel +49/241/8021501 | Fax +49/241/8022321 | http://dbis.rwth-aachen.de/c

OWL Ontologies

- What's inside an OWL ontology
  - Classes + class-hierarchy
  - Properties (Slots) / values
  - Relations between classes (inheritance, disjoints, equivalents)
  - Restrictions on properties (type, cardinality)
  - Characteristics of properties (transitive, …)
  - Annotations
  - Individuals

- Reasoning tasks: classification, consistency checking

Working with OWL syntax

```
<owl:Class rdf:ID="Virus">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  ></rdfs:comment>
  <owl:disjointWith>
    <owl:Class rdf:ID="Bacterium"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="MicroOrganism"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Virus</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Bacterium">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#MicroOrganism"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Bacterium</rdfs:label>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  ></rdfs:comment>
</owl:Class>
<owl:Class rdf:about="#MicroOrganism">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:ID="Organism"/>
        <owl:Restriction>
          <owl:someValuesFrom>
            <owl:Class rdf:ID="MicroScale"/>
          </owl:someValuesFrom>
          <owl:onProperty>
            <owl:ObjectProperty rdf:ID="asScaleRealm"/>
          </owl:onProperty>
```

RWTH
Tel +49

OWL Syntax: Abstract Syntax

- One of the clearer human-readable syntaxes

```
Class(SpicyPizza complete
  annotation(rdfs:label "PizzaTemperada"@pt)
  annotation(rdfs:comment "Any pizza that has a      spicy topping is a
SpicyPizza"@en)
  Pizza
  restriction(hasTopping someValuesFrom(SpicyTopping))
)
```

OWL Syntax: N3

- Recommended for human-readable fragments

```
default:SpicyPizza
    a owl:Class ;
    rdfs:comment "Any pizza that has a spicy topping is a
     SpicyPizza"@en ;
    rdfs:label "PizzaTemperada"@pt ;
    owl:equivalentClass
         [ a owl:Class ;
           owl:intersectionOf (default:Pizza [ a owl:Restriction ;
                   owl:onProperty default:hasTopping ;
                   owl:someValuesFrom default:SpicyTopping
                  ])
         ] .
```

OWL Syntax: RDF/XML

- Recommended for serialisation

```
<owl:Class rdf:ID="SpicyPizza">
  <rdfs:label xml:lang="pt">PizzaTemperada</rdfs:label>
  <rdfs:comment xml:lang="en">Any pizza that has a spicy topping is a
  SpicyPizza</rdfs:comment>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Pizza"/>
        <owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#hasTopping"/>
          </owl:onProperty>
          <owl:someValuesFrom rdf:resource="#SpicyTopping"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Open world vs. Closed world

- Open world: A statement can be true unless explicitly known to be false.
  - What is not known/stated can be true.

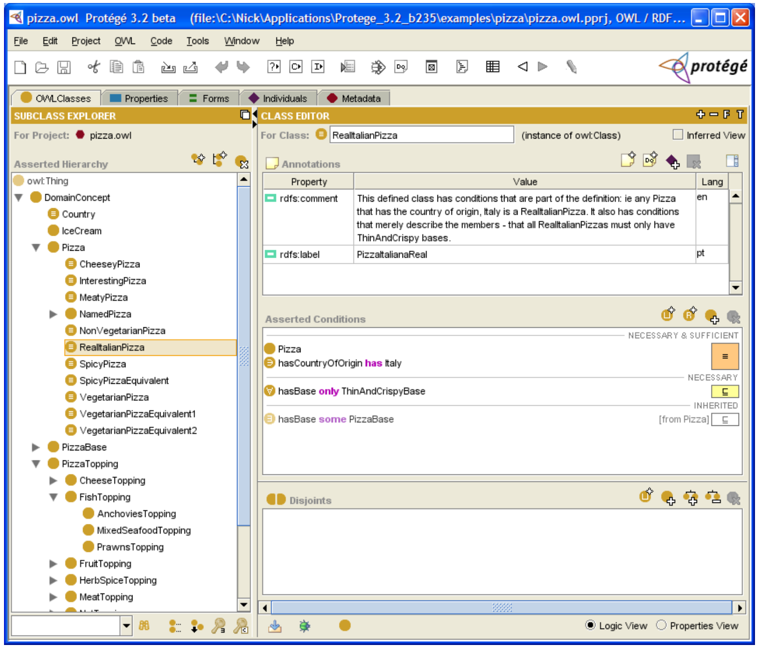
- Closed world: A statement is false unless explicitly known to be true.
  - What is not known is assumed to be false.

TOOLS

Tool Support for OWL

- Ontology editors
  - Protégé (http://protege.stanford.edu/)
  - OilED (http://oiled.man.ac.uk/)
  - OntoStudio (http://www.ontoprise.de/en/home/products/ontostudio/)
- APIs
  - OWL-API (http://owlapi.sourceforge.net/)
  - Jena (http://jena.sourceforge.net/)
- Reasoners
  - Hoolet (http://owl.man.ac.uk/hoolet/)
  - Fact++ (http://owl.man.ac.uk/factplusplus/)
  - KAON2 (http://kaon2.semanticweb.org/)
  - Pellet (http://clarkparsia.com/pellet/)

- Developed by Stanford Medical Informatics
- Has a large user community (approx 30k)
- Support
  - Graph view, consistency check, web, merging
- No support
  - Addition of new basic types
  - Limited multi-user support



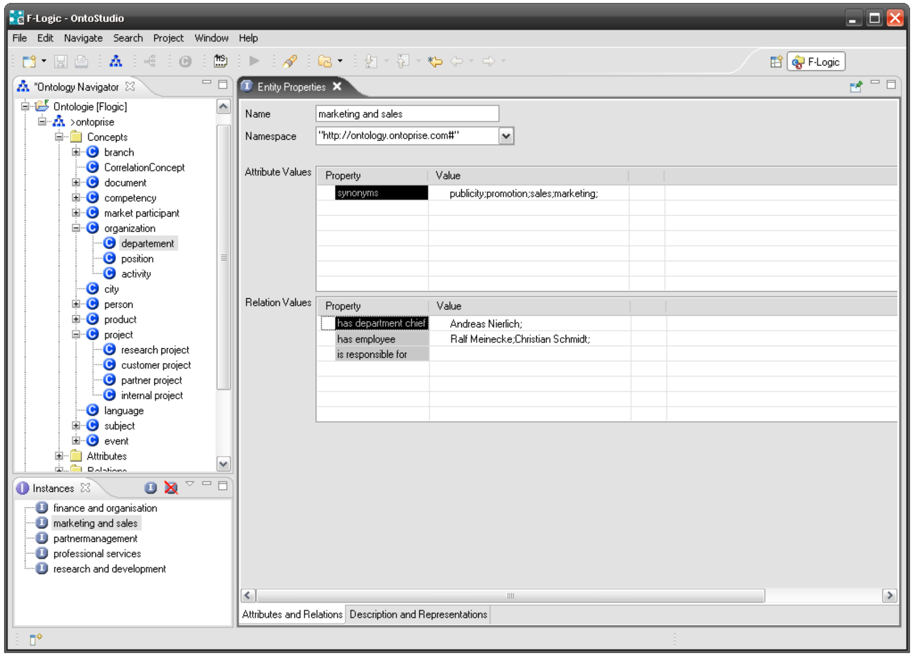
RWTH Informatik 5 | Ahornstr. 55 D-52056 Aachen Germany |
Tel +49/241/8021501 | Fax +49/241/8022321 | http://dbis.rwth-aachen.de/cms

OilEd

- Developed by Information Management Group, CS Dept., Univ. of Manchester, UK
- Simple editor, not intended as a full ontology development environment
- Support
  - Consistency check, web
- No support
  - Graph view, extensibility

OntoStudio

- Developed by Ontoprise, Germany
- Support
  - Graph view, consistency check, web
  - Built-in inference engine, DBMS, collaborative working and ontology library



RWTH Informatik 5 | Ahornstr. 55 D-52056 Aachen Germany |
Tel +49/241/8021501 | Fax +49/241/8022321 | http://dbis.rwth-aachen.de/cms

OWL-API

- A Java interface and implementation for the W3C Web Ontology Language OWL
- Open source and is available under the LGPL License
- Support
  - OWL 2
  - RDF/XML parser and writer
  - OWL/XML parser and writer
  - OWL Functional Syntax parser and writer
  - Integration with reasoners such as Pellet and FaCT++

Jena

- A Java framework for building Semantic Web applications
- Jena is open source
- Initiated by Hewlett Packard (HP) Labs Semantic Web Programme.
- Support:
  - A RDF API
  - Reading and writing RDF in RDF/XML, N3 and N-Triples
  - An OWL API
  - In-memory and persistent storage
  - SPARQL query engine

Hoolet

- An implementation of an OWL-DL reasoner
- Uses a first order prover.
- The ontology is translated to collection of axioms (in an obvious way based on the OWL semantics) and this collection of axioms is then given to a first order prover for consistency checking.

KAON 2

- An infrastructure for managing OWL-DL, SWRL, and F-Logic ontologies
- Joint effort of: Research Center for Information Technologies, University of Karlsruhe, University of Manchester
- Support
  - An API for programmatic management of OWL-DL, SWRL, and F-Logic ontologies,
  - A stand-alone server providing access to ontologies in a distributed manner using RMI,
  - An inference engine for answering conjunctive queries (expressed using SPARQL syntax),
  - A DIG interface, allowing access from tools such as Protégé,
  - A module for extracting ontology instances from relational databases.

Pellet

- Open-source Java OWL DL reasoner
- Support expressivity of SROIQ(D)
- Supports SWRL rules
- Available through AGPL version 3 licence

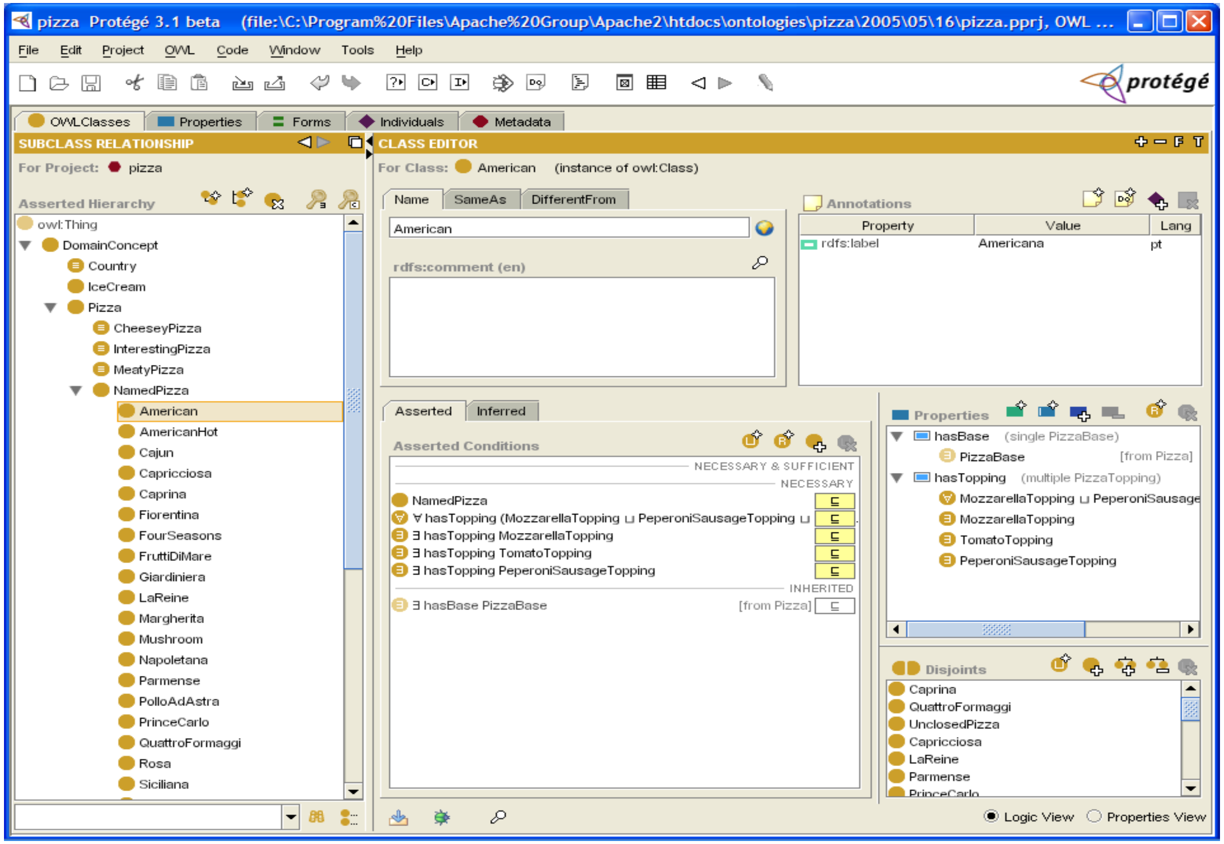
FaCT++

- New generation and C++ implementation of FaCT
- Support expressivity of SROIQ(D)
- No support for Rules
- Available through GNU public license
- Integrated in Protege 4.0

Basic Protégé-OWL usage

Protégé OWL: a GUI environment

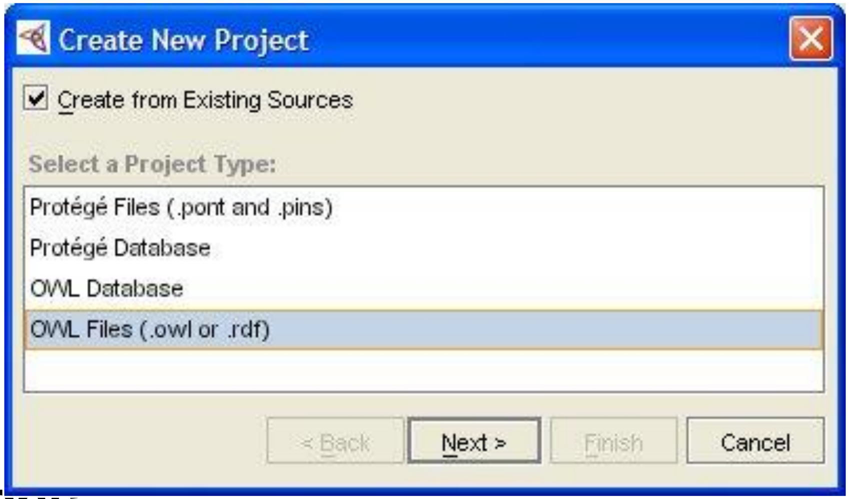
- OWL environment within PROTÉGÉ framework
- Most widely used tool for editing and managing OWL ontologies
- Approx 166,000 registered users



Protégé OWL features

- Loading and saving OWL files & databases
- Graphical editors for class expressions
- Access to description logics (DL) reasoners via Protégé GUI and the DIG interface
- Ontology visualization plug-ins
- Built on Protégé platform
  - Can hook in custom-tailored components
  - Can serve as API for new applications (including web applications)

Loading OWL files



1. *If you only have an OWL file:*
   - ***File → New Project***
   - *Select **OWL Files** as the type*
   - *Tick **Create from existing sources***
   - ***Next** to select the .owl file*

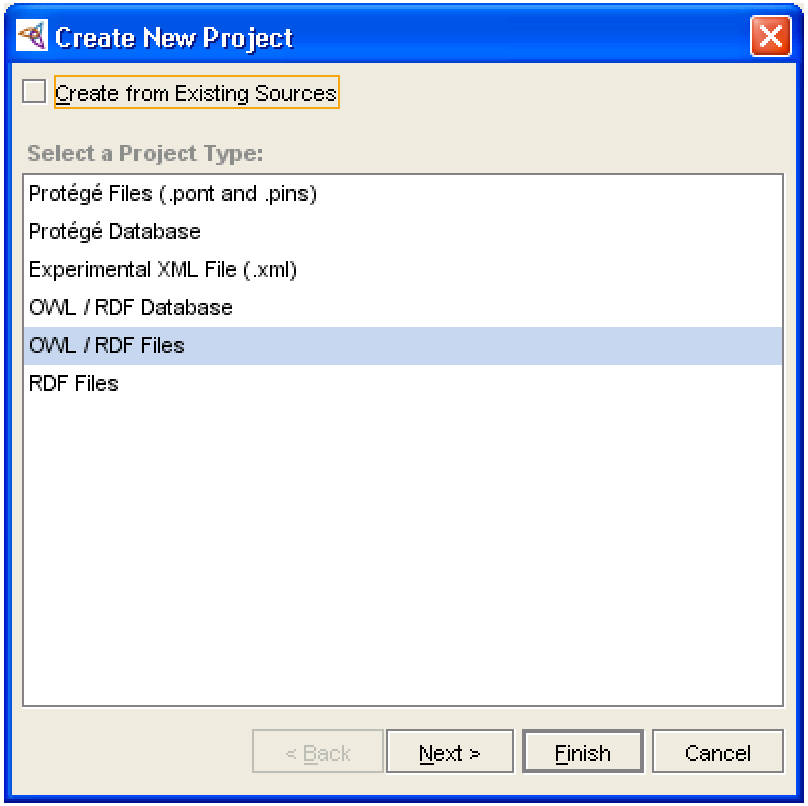
2. *If you've got a valid project file*:*
   - ***File → Open Project***
   - *select the .pprj file*

Create or load an OWL project

File → New Project

OR

File → Open Project



Protégé OWL Overview

Classes
- Subclass relationships
- Disjoint classes

Properties
- Characteristics (transitive, inverse)
- Range and Domain
  - ObjectProperties (references)
  - DatatypeProperties (simple values)

Individuals
- Property values

Class Descriptions
- Restrictions
- Logical expressions

OWL for data exchange

OWL for classification and reasoning

Ontology Development Process

- Steps:
  - Consider reuse
  - Enumerate terms
  - Define classes
  - Define properties
  - Define constraints
  - Create instances

In reality - an iterative process

Establish Purpose

- Determine Scope:

- What will the ontology be used for?

Classification of Pneumonia:

  - Bacterial Pneumonia (caused by bacteria)
  - Pneumococcal Pneumonia (caused by a particular kind of bacteria)
  - Viral Pneumonia (caused by viruses)
  - Mixed Pneumonia (caused by both bacteria and viruses)

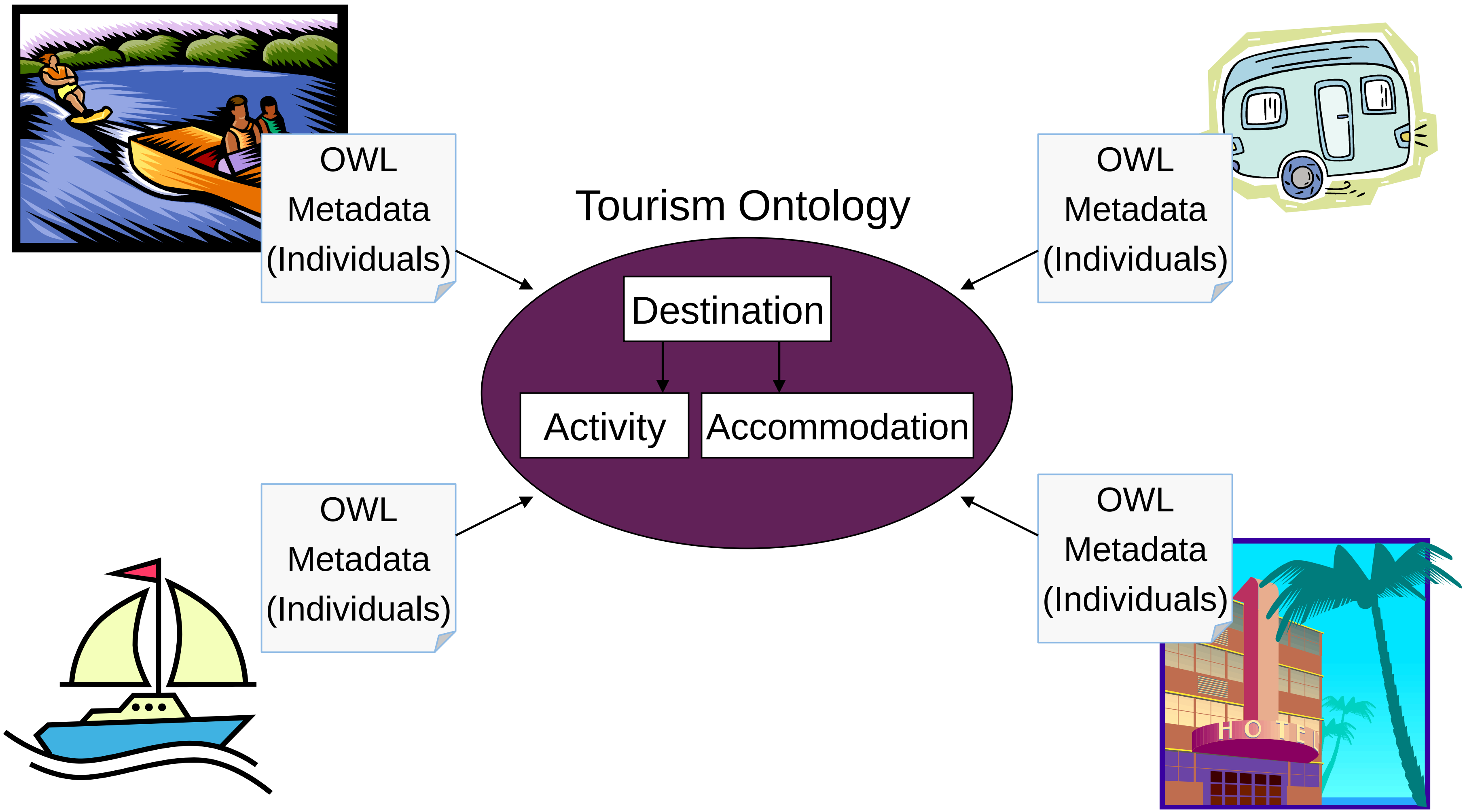
Enumerate Important Entities

- What are the entities we need to talk about?
  Pneumonias, infectious organisms.
- What are the properties of these entities?
  hasRadiologyFinding, hasLocus, hasCause.
- What do we want to say about the entities?
  Pneumonias cause radiology opacity findings
  Pneumonias are located in lung
  Mixed pneumonias are caused by bacteria and viruses.
  …

RWTH Informatik 5 | Ahornstr. 55 D-52056 Aachen Germany |
Tel +49/241/8021501 | Fax +49/241/8022321 | http://dbis.rwth-aachen.de/cms

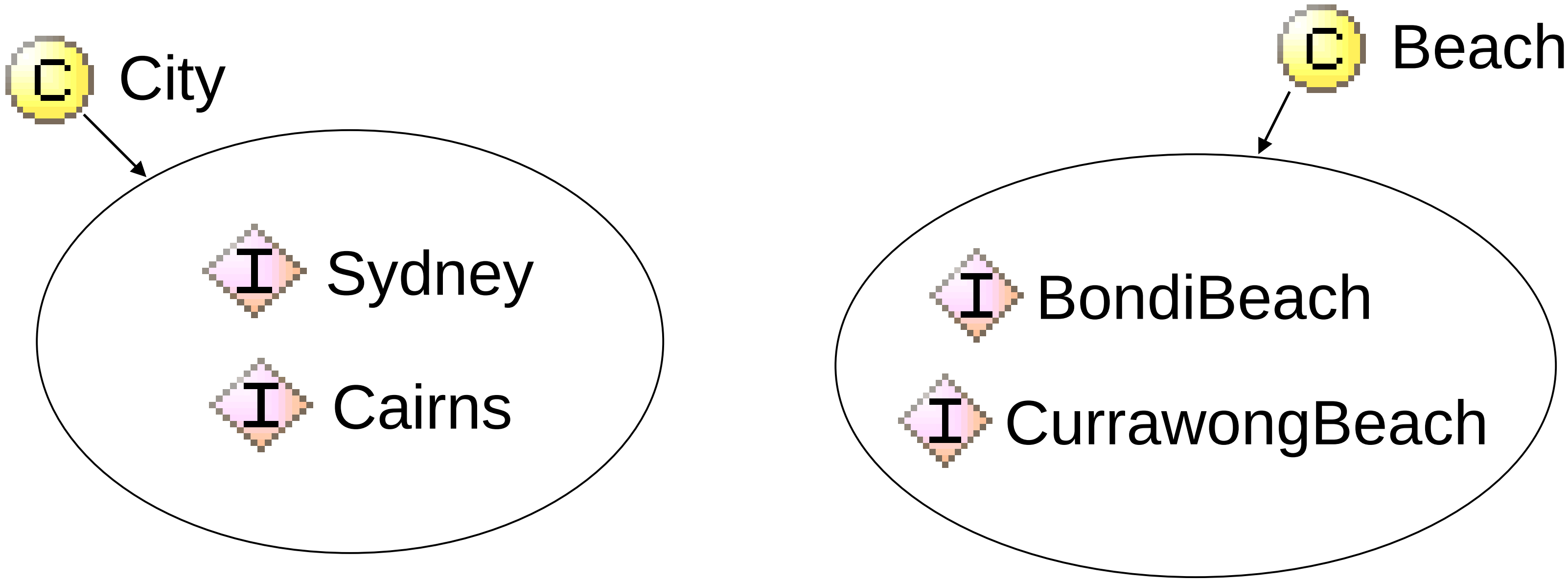
CLASSES (Types, Universals)

Tourism Example



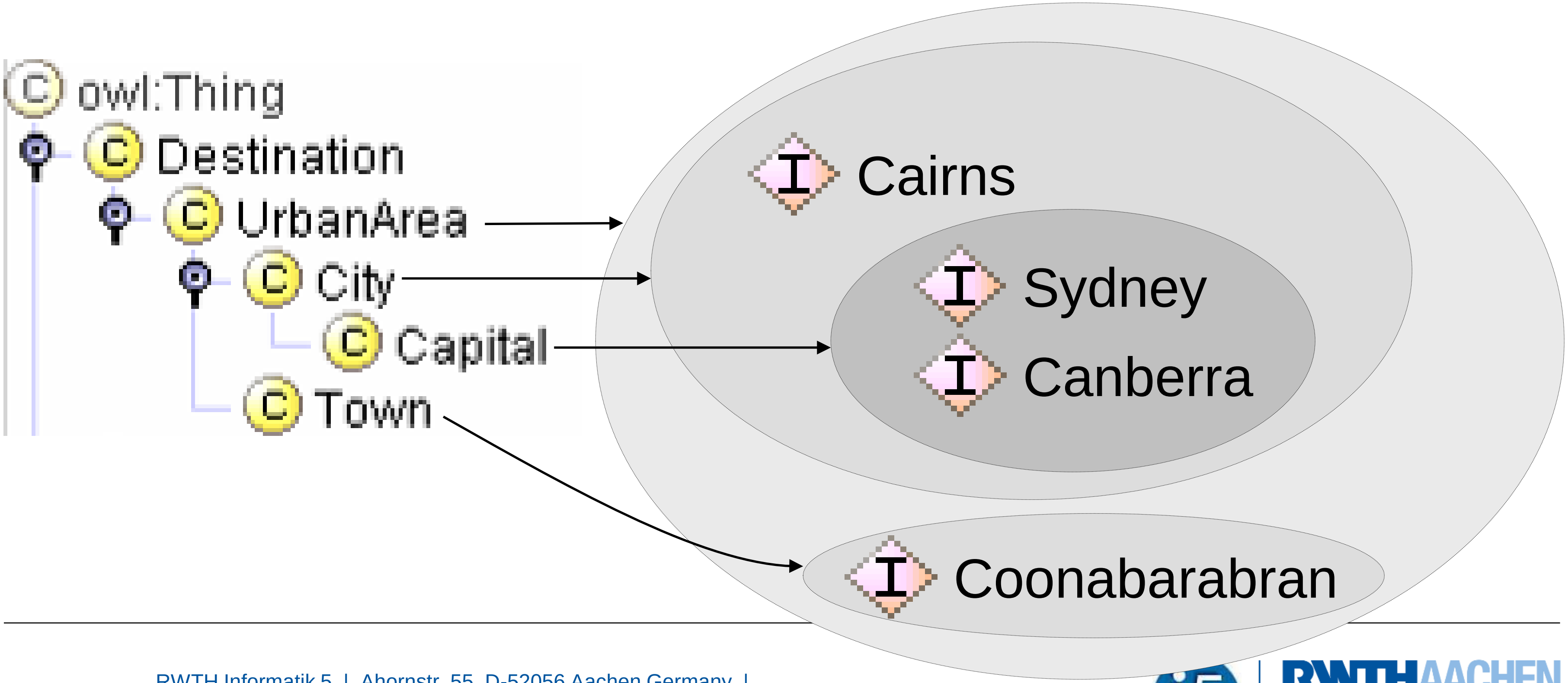
Classes

- Sets of **individuals** with common characteristics
- **Individuals** are instances of at least one class



Superclass Relationship

- Classes organized in a hierarchy implies subsumption
- Direct instances of subclass are also (indirect) instances of superclasses



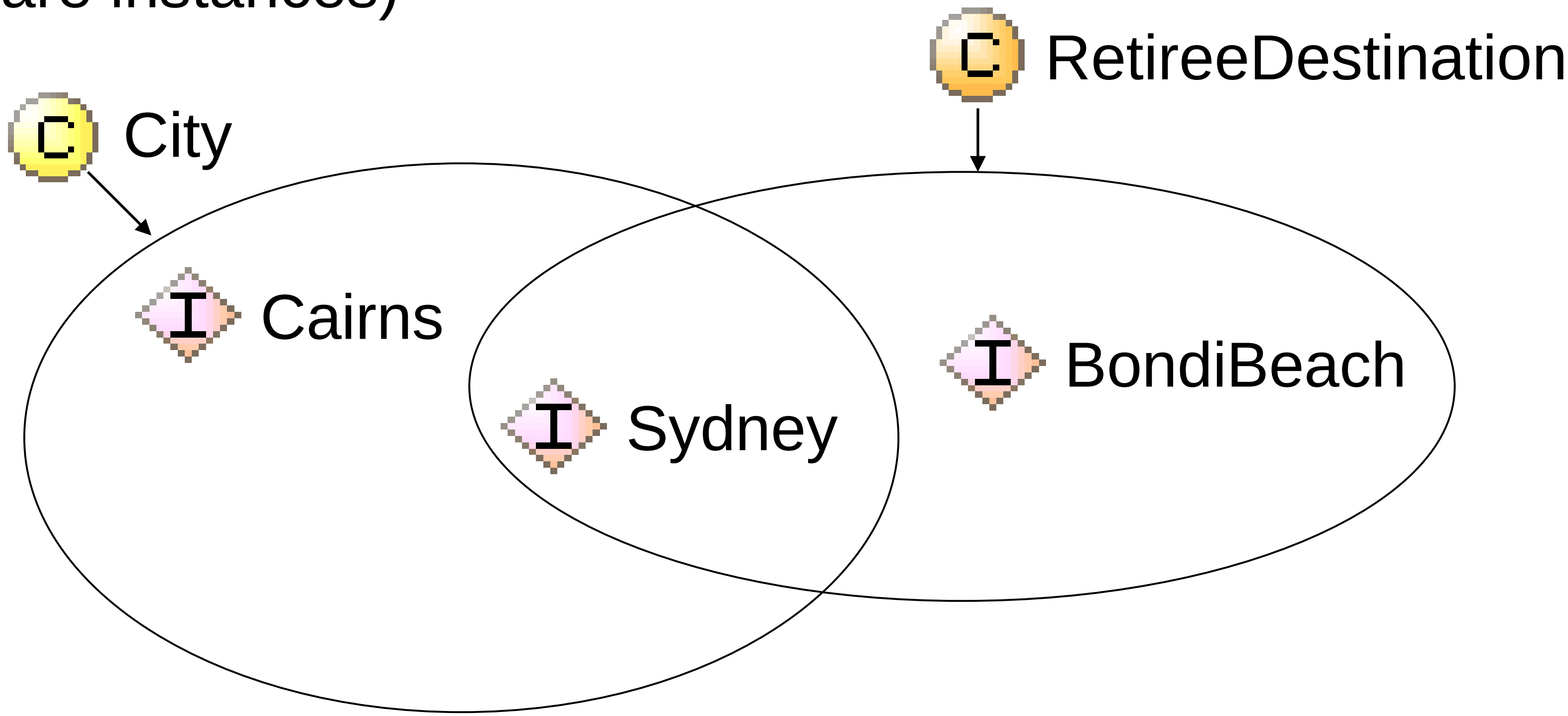
Example Subclasses

```
<owl:Class rdf:ID="PotableLiquid">
   <rdfs:subClassOf  rdf:resource="#ConsumableThing" /> …
</owl:Class>


<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
   <rdfs:label xml:lang="en">wine</rdfs:label>
   <rdfs:label xml:lang="fr">vin</rdfs:label> ...
</owl:Class>
```

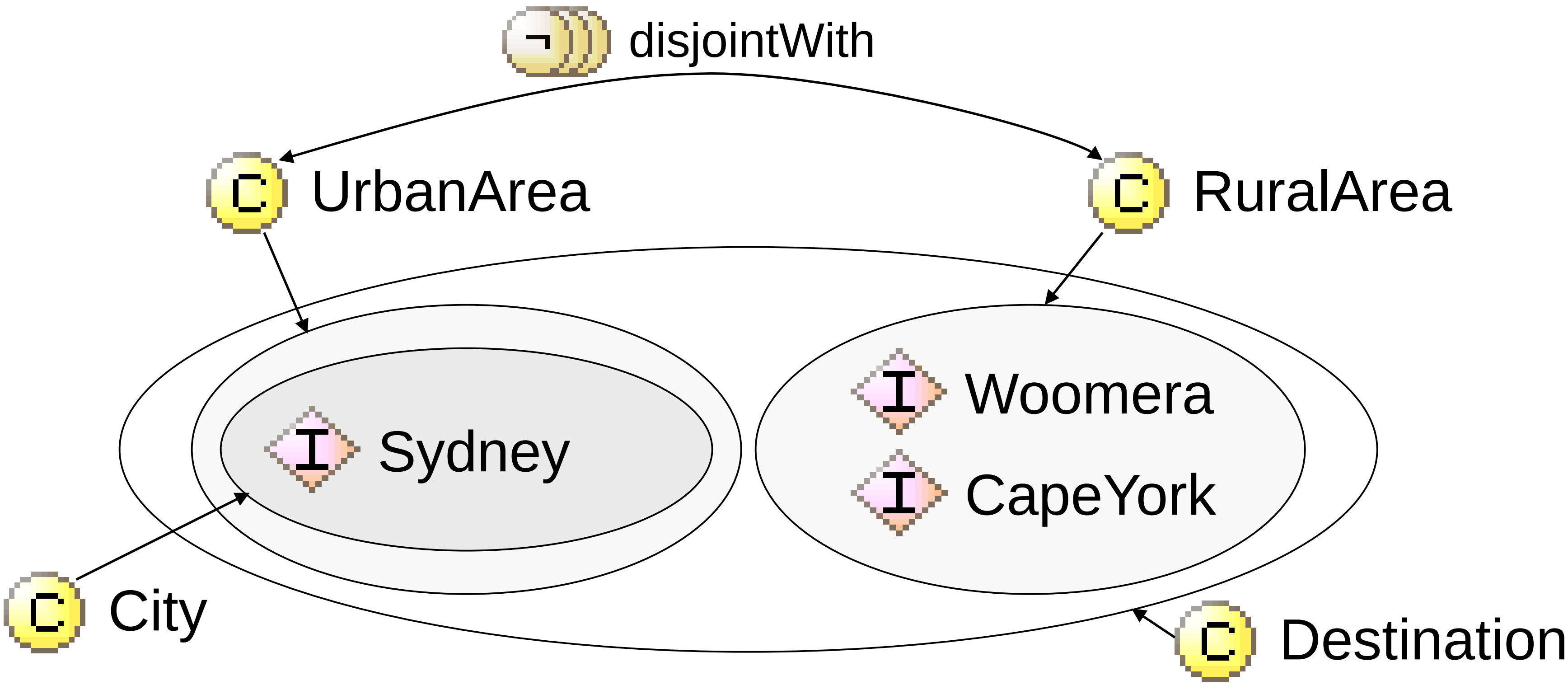
Class overlap

- Classes can overlap arbitrarily
- Classes are assumed non-disjoint by default (ie, they may share instances)



Class Disjointness

- All classes could potentially overlap
- Specify **disjointness** to make sure they don't share instances



Example disjoint

```
<owl:Class rdf:about="#Man"> <owl:disjointWith
  rdf:resource="#Woman"/>
</owl:Class>
```

only in OWL full !

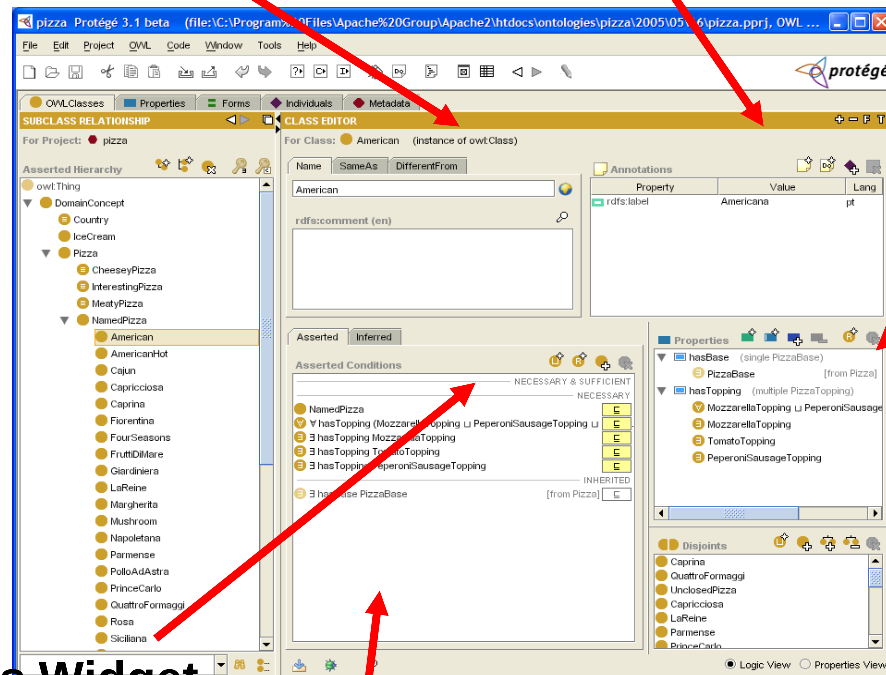
Class Editor

Class annotations (for class metadata)

Class name and documentation

Properties "available" to Class

Disjoints widget



Conditions Widget

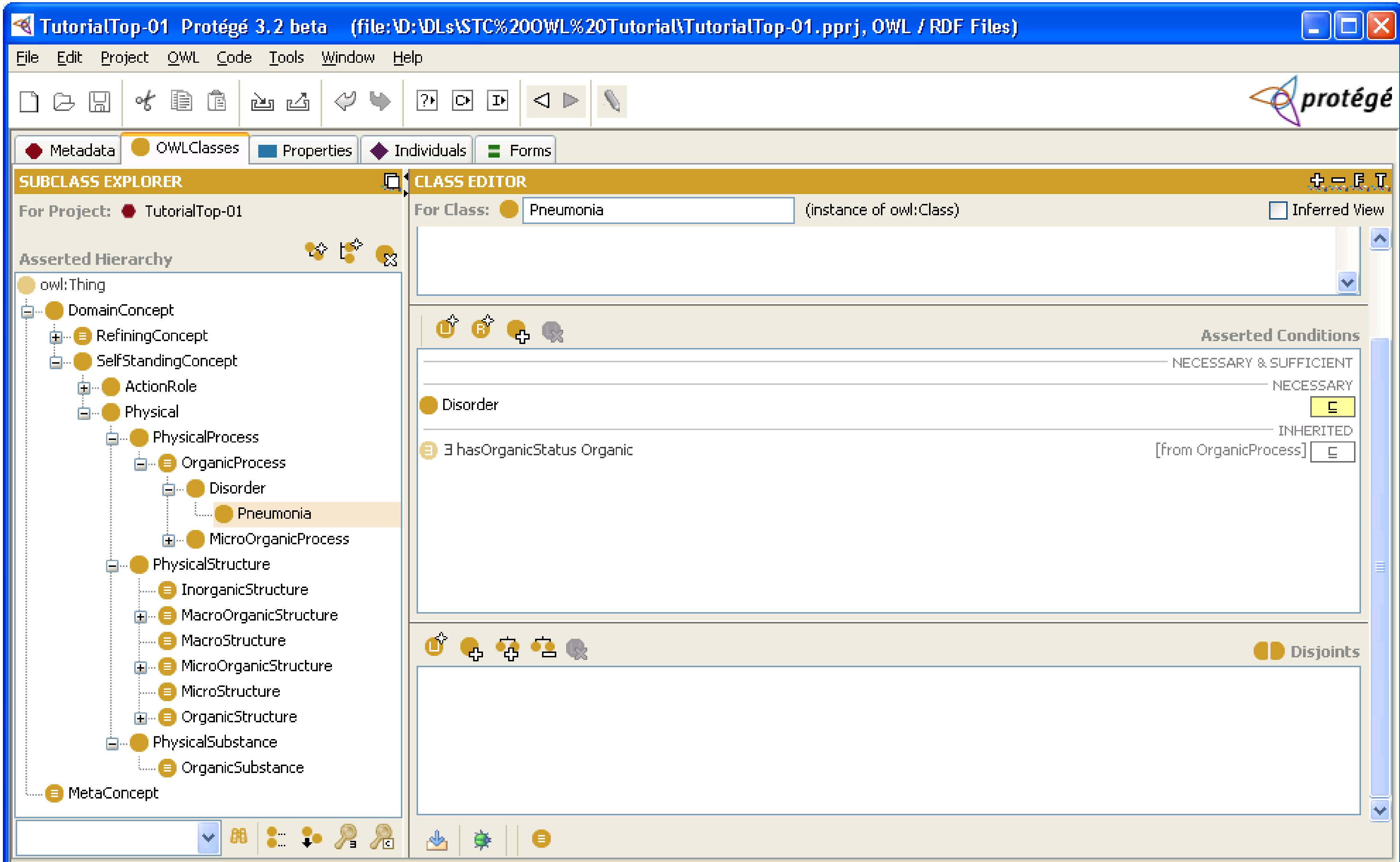
Class-specific tools (find usage etc)

RWTH Informatik 5 | Ahornstr. 55 D-52056 Aachen Germany |
Tel +49/241/8021501 | Fax +49/241/8022321 | http://dbis.rwth-aachen.de/cms

Define classes and the class hierarchy

- Identify Classes (from the previous entity list)
  - If something can have a kind then it is a Class
  - “Kind of Pneumonia” √ - Pneumonia is a Class
  - “Kind of Bacteria” √ Bacteria is a Class
- Arrange Classes in an hierarchy
  - PneumococcalPneumonia is a subclass of Pneumonia
    - Every PneumococcalPneumonia is a Pneumonia
  - Pneumococcus is a subclass of Bacteria
    - Every Pneumococcus is a Bacteria
  - MixedPneumonia is a subclass of Pneumonia
    - Every MixedPneumonia is a Pneumonia

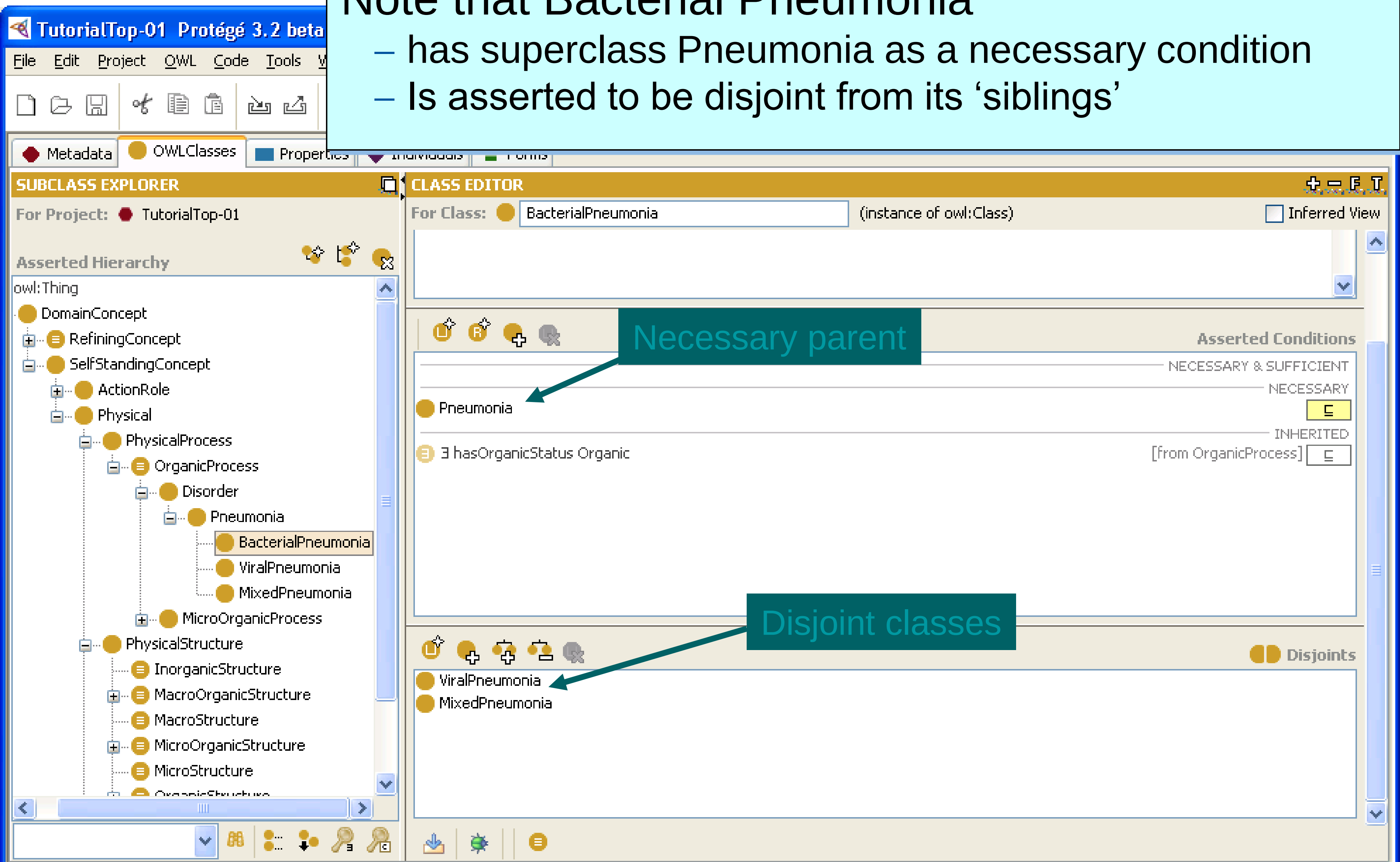
Create classes: “Pneumonia” class



Class Disjoints

Note that Bacterial Pneumonia

- has superclass Pneumonia as a necessary condition
- Is asserted to be disjoint from its 'siblings'

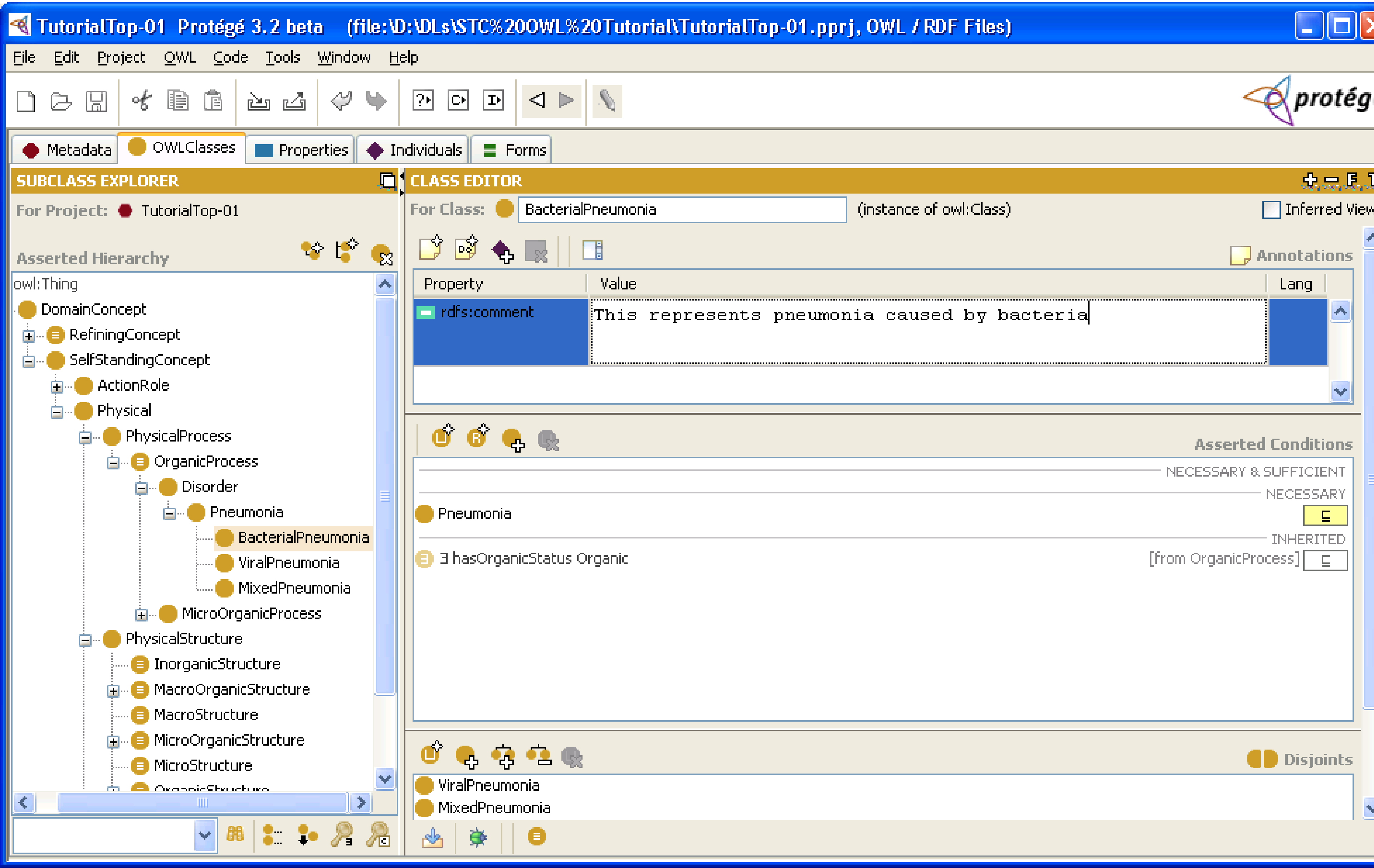


What it means

- All BacterialPneumonias are Pneumonias
  - **No BacterialPneumonia is not a Pneumonia**
- Nothing is both:
  - **a BacterialPneumonia and a ViralPneumona**
  - **a BacterialPneumonia and a MixedPneumonia**

NB: In OWL classes ***can overlap*** unless declared disjoint!

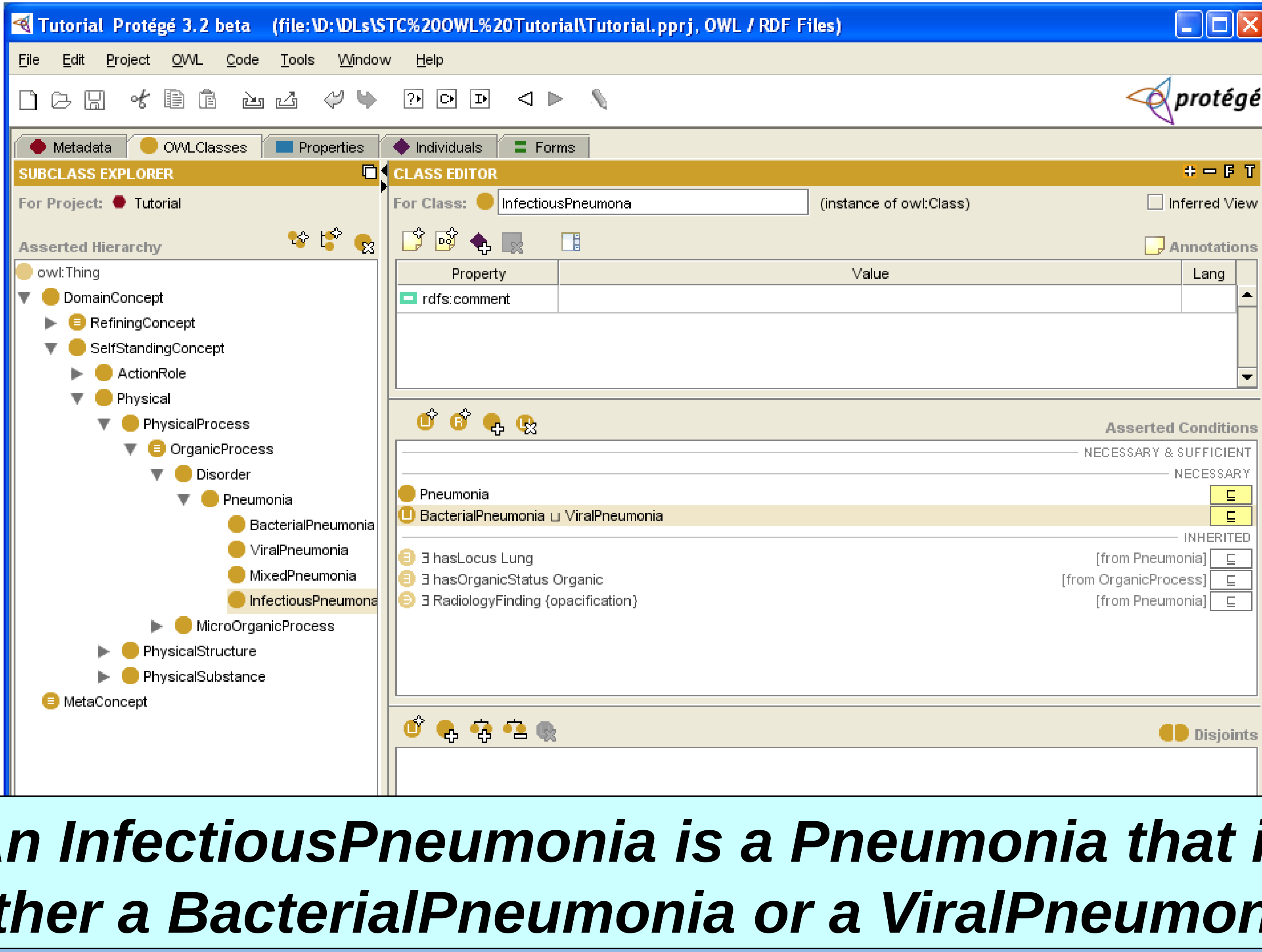
Add metadata on Classes



Another Way to Create Classes

- A class can be the **union** of two classes
  - An InfectiousPneumonia is either a BacterialPneumonia or a ViralPneumonia
- A class can be the **intersection** of two classes
  - A MixedPneumonia is any Pneumonia that is caused by both Bacteria and Viruses
- A class can be the **complement** of another class
  - Noninfectious pneumonia is any pneumonia that is not caused by an infectious agent (bacteria or virus)

Create a class by composition



An InfectiousPneumonia is a Pneumonia that is either a BacterialPneumonia or a ViralPneumonia