

# Developing Biomedical Ontologies in OWL

**Alan Rector**

School of Computer Science / Northwest Institute of Bio-Health Informatics  
rector@cs.man.ac.uk

with special acknowledgement to Jeremy Rogers

[www.co-ode.org](http://www.co-ode.org)

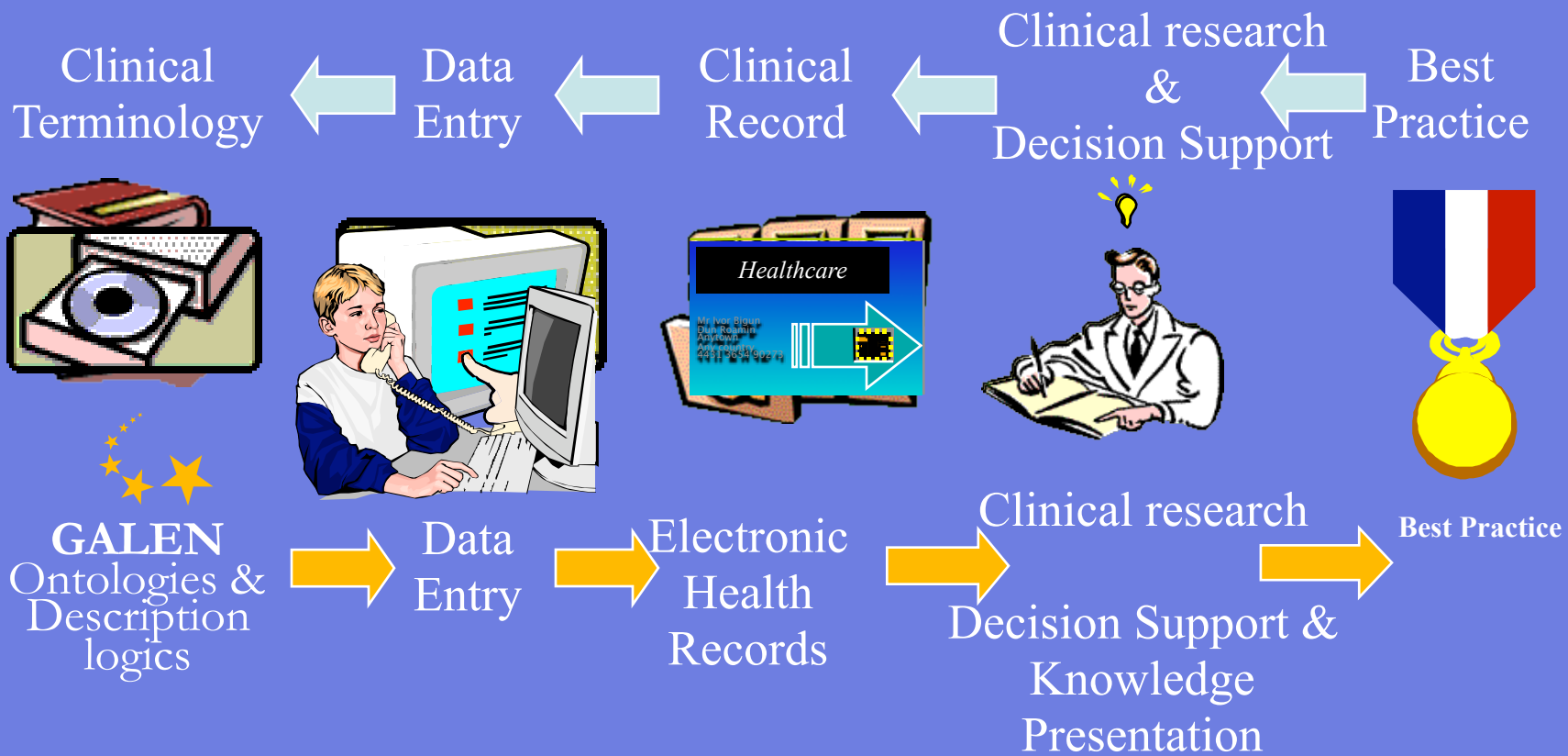
[www.clinical-escience.org](http://www.clinical-escience.org)

[www.opengalen.org](http://www.opengalen.org)

# Tools and downloads

- Protege4Alpha
  - [protege.stanford.edu](http://protege.stanford.edu)
  - <http://protege.stanford.edu/download/prerelease-alpha/protege-bin-4.0-alpha.zip>
- GraphViz - required for OWLViz
  - <http://www.graphviz.org/>
- Tutorial handouts and Ontologies
  - <http://www.cs.man.ac.uk/~rector/tutorials/Medinfo-2007>
- Preparatory material
  - If you haven't done so already, please read the introduction to OWL and Protege-OWL
  - <http://www.co-ode.org/resources/tutorials/protege-owl-tutorial.php>

# Where I come from



# “Ontologies” in Information Systems

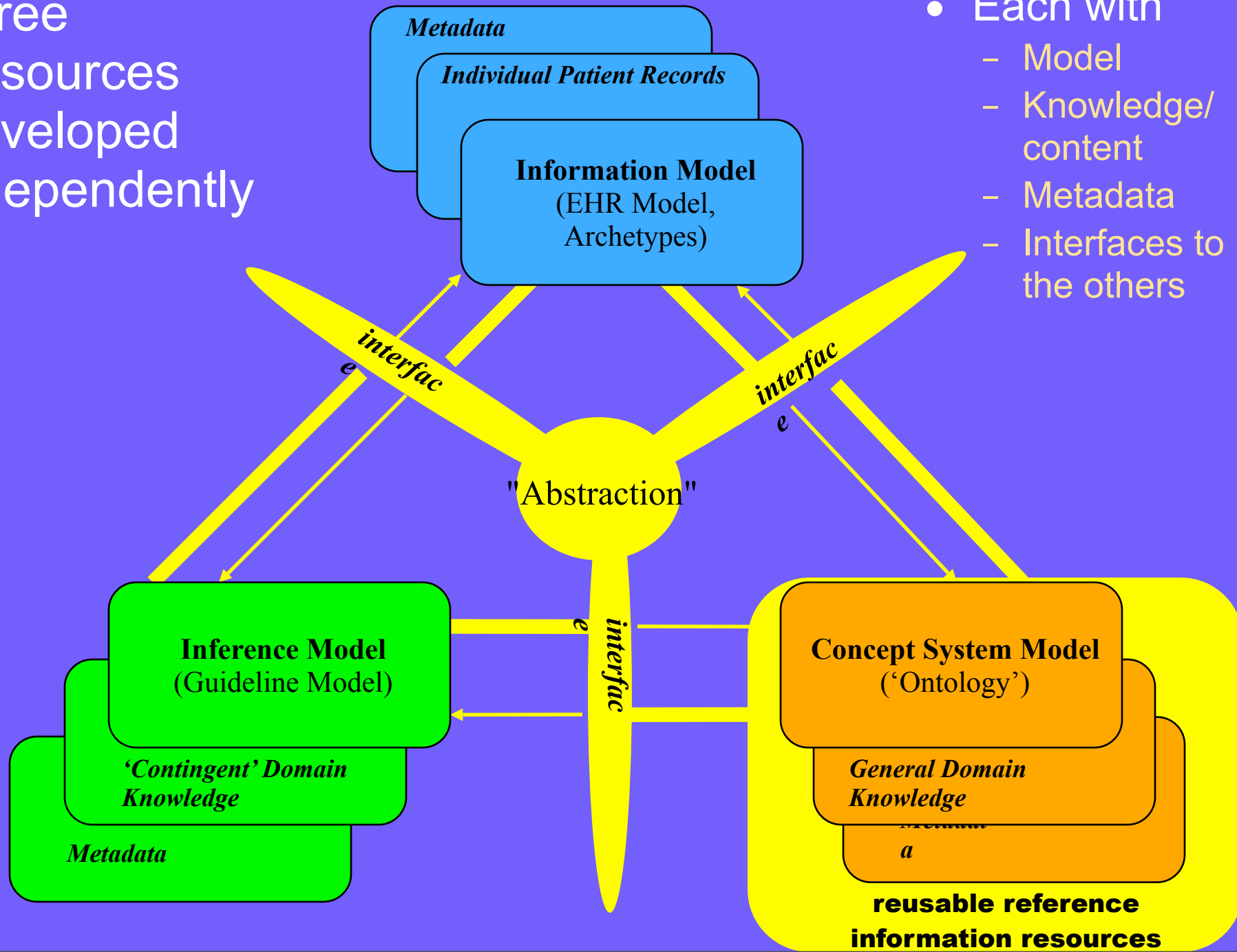
- What information systems can say and how - “Models of Meaning”
  - Mathematical theories - although usually weak ones
    - evolved at the same time as Entity Relation and UML style modelling
- Managing Scalability / complexity - “Knowledge driven systems”
  - Housekeeping tools for expert systems
    - Organising complex collections of rules, forms, guidelines, ...
- Interoperability
  - The common grounding information needed to achieve communication
  - Standards and terminology
- Communication with users
  - Document design decisions
- Testing and quality assurance
  - sufficient constraints to know when it breaks
  - Empower users to make changes safely
- ... but ***“They don’t make the coffee”***
  - ***just one component of the system / theory***

# My definition of an ontology

- Short version:  
*“a representation of the shared background knowledge for a community”*
- Long version:  
*“an implementable model of the entities that need to be understood in common in order for some group of software systems and their users to function and communicate at the level required for a set of tasks”*
- ... and “it doesn’t make the coffee”  
*Just one of at least three components of a complete system*

Three Resources Developed independently

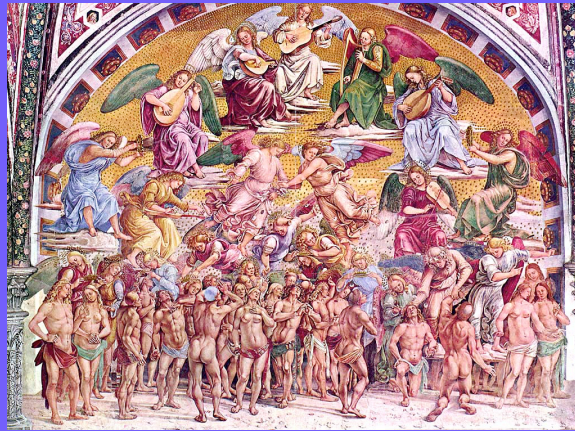
- Each with
  - Model
  - Knowledge/content
  - Metadata
  - Interfaces to the others



# By way of User Centred Design Knowledge Representation / ontologies was a solution, not a goal

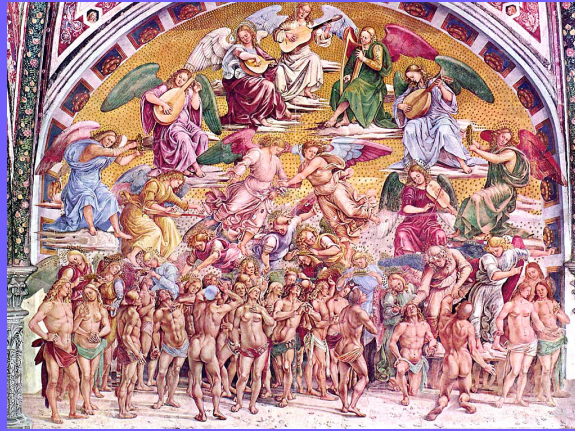






- Solution space
  - Ontologies
  - Information Models
  - Logics
  - Rules
  - Frames
  - Planners
  - Logic programming
  - Bayes nets
  - Decision theory
  - Fuzzy sets
  - Open / closed world
- Problem space
  - Answer questions
  - Advising on actions
  - Hazard monitoring
  - Creating forms
  - Discovering resources
  - Constraint actions
  - Assess risk
  - ...





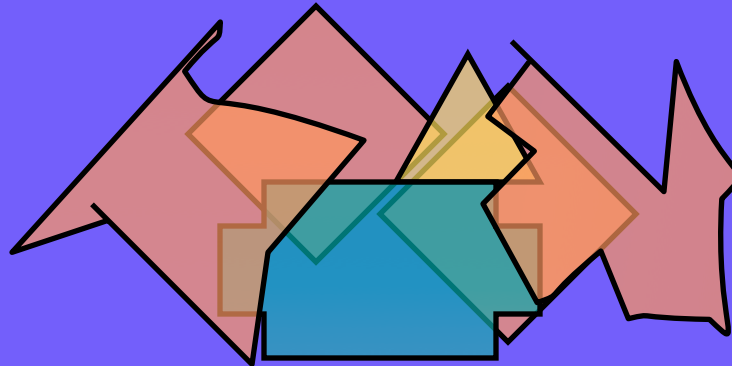
- Problem space
  - Answer questions
  - Advising on actions
  - Hazard monitoring
  - Creating forms
  - Discovering resources
  - Constraint actions
  - Assess risk
  - ...



- Solution space
  - Ontologies
  - Information Models
  - Logics
  - Rules
  - Frames
  - Planners
  - Logic programming
  - Bayes nets
  - Decision theory
  - Fuzzy sets

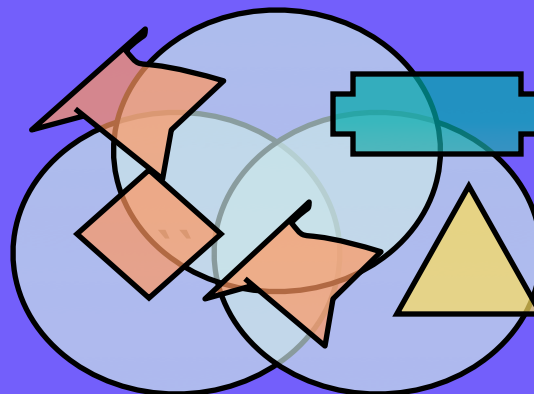
# Problem space & solution space

Problem space



Guidelines, Patterns, Tools

Solution space



# Topics for today

- Motivation
- Very Brief Review of OWL - a naive version of “pneumonia”
- Normalisation & Why Classify?
  - Why use a computable subset of logic?
- Modularisation - doing it in layers
- Anatomy, parts and Disorders - a less naive version of “pneumonia”
  - Pneumonitis and pneumonias
  - A disorders of the lung
- Quantities and Units - if time
- Normal, NonNormal & Pathological
  - Using negation
- Summary

# Why use a Classifier?

- To *compose* concepts
  - *Allow conceptual lego*
- To avoid *combinatorial explosions*
  - *Keep bicycles from exploding* To manage *polyhierarchies*
  - *Adding abstractions (“axes”) as needed*
  - *Normalisation*
    - *Untangling*
      - *labelling of “kinds of is-a”*
- To manage *context*
  - *Cross species, Cross disciplines, Cross studies*
- To check *consistency* and *help users find errors*

# Assertion:

The arrival of computable logic-based ontologies/OWL gives new opportunities to make ontologies more manable and modular

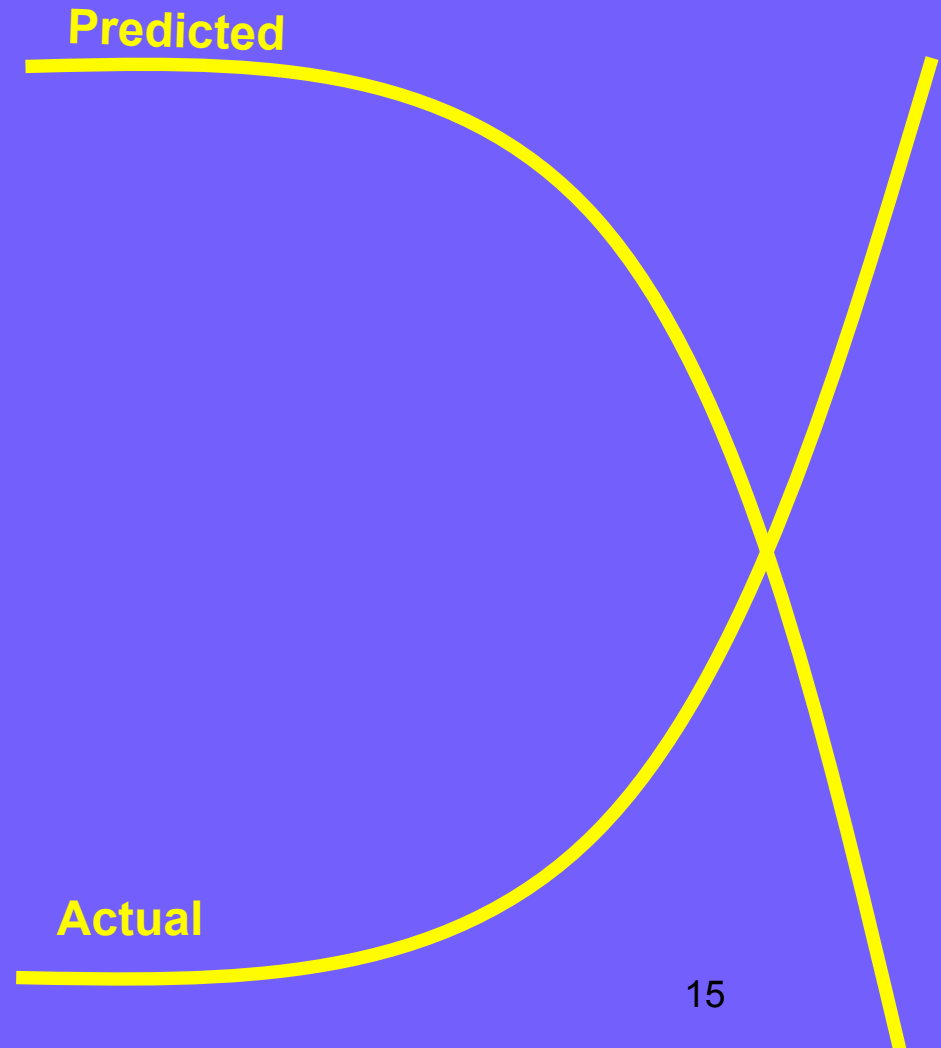
- ▶ Let the ontology authors
  - ▶ create discrete modules
  - ▶ describe the links between modules
- ▶ Let the logic reasoner
  - ▶ Organise the result
- ▶ Let users see the consequences of their actions
  - ▶ Very few people can do logic well
    - ▶ And almost none quickly

Fundamental problems:

Enumeration doesn't scale

# The scaling problem: The combinatorial explosion

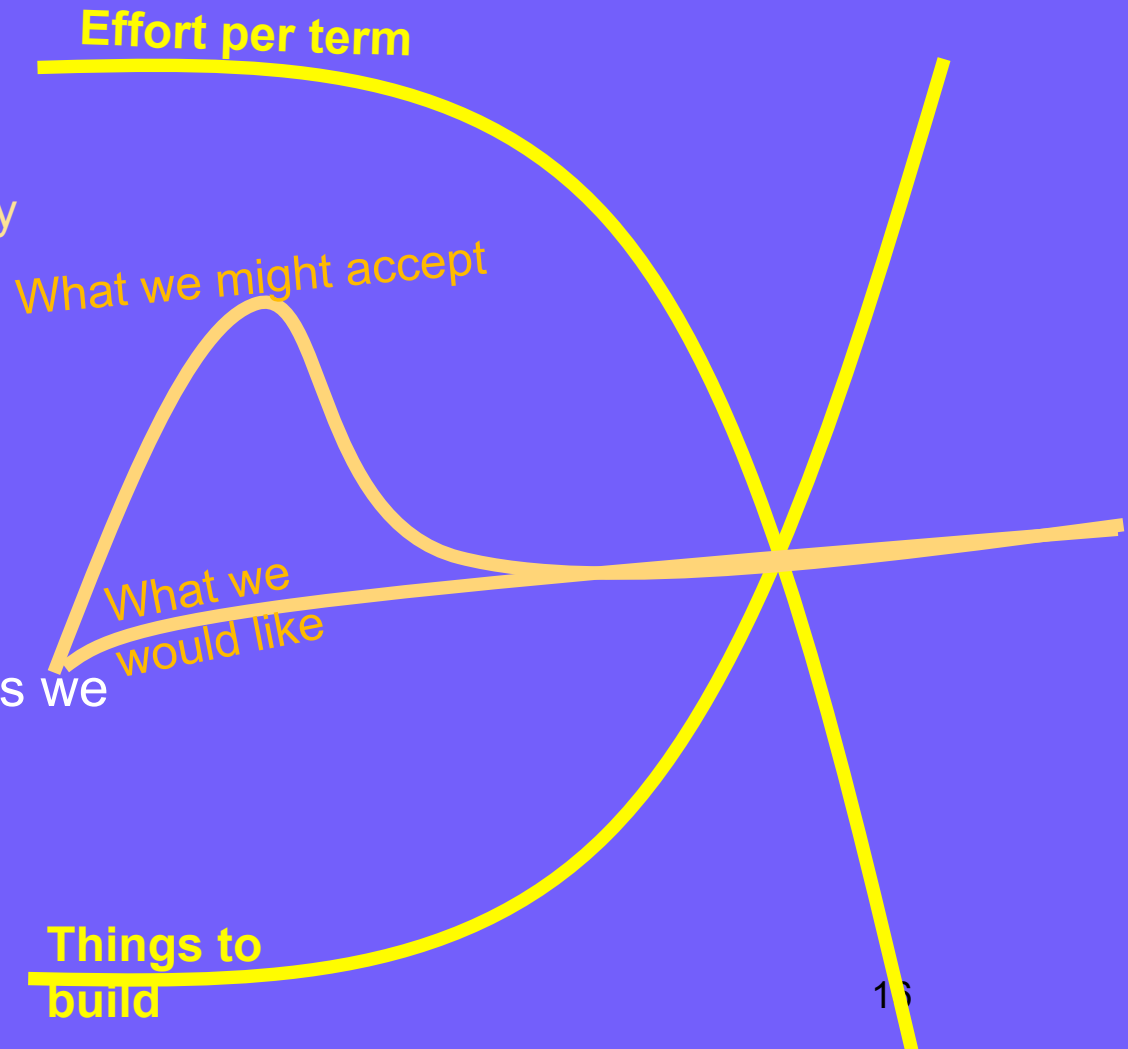
- It keeps happening!
  - “Simple” brute force solutions do not scale up!
- Conditions x sites x modifiers x activity x context→
  - *Huge number of terms to author*
  - *Software CHAOS*





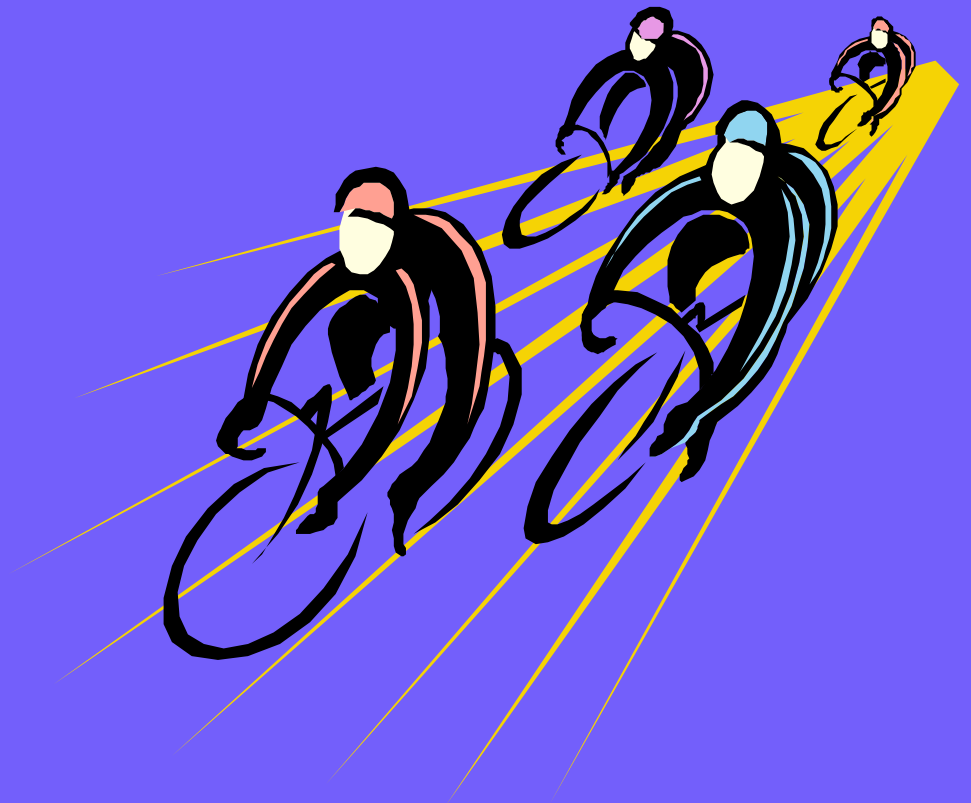
# Combination of things to be done & time to do each thing

- Terms and forms needed
  - Increases exponentially
- Effort per term or form
  - Must decrease to compensate
- To give the effectiveness we want
  - Or might accept



# The exploding bicycle

- 1972 ICD-9 (E826) 8
- READ-2 (T30..) 81
- READ-3 87
- 1999 ICD-10 .....





# Defusing the exploding bicycle:

## 500 codes in pieces

- 10 things to hit...
  - Pedestrian / cycle / motorbike / car / HGV / train / unpowered vehicle / a tree / other
- 5 roles for the injured...
  - Driving / passenger / cyclist / getting in / other
- 5 activities when injured...
  - resting / at work / sporting / at leisure / other
- 2 contexts...
  - In traffic / not in traffic

V12.24 Pedal cyclist injured in collision with two- or three-wheeled motor vehicle, unspecified pedal cyclist, nontraffic accident, while resting, sleeping, eating or engaging in other vital activities

# Conceptual Lego... it could be...

Structured Data Entry

File Edit Help

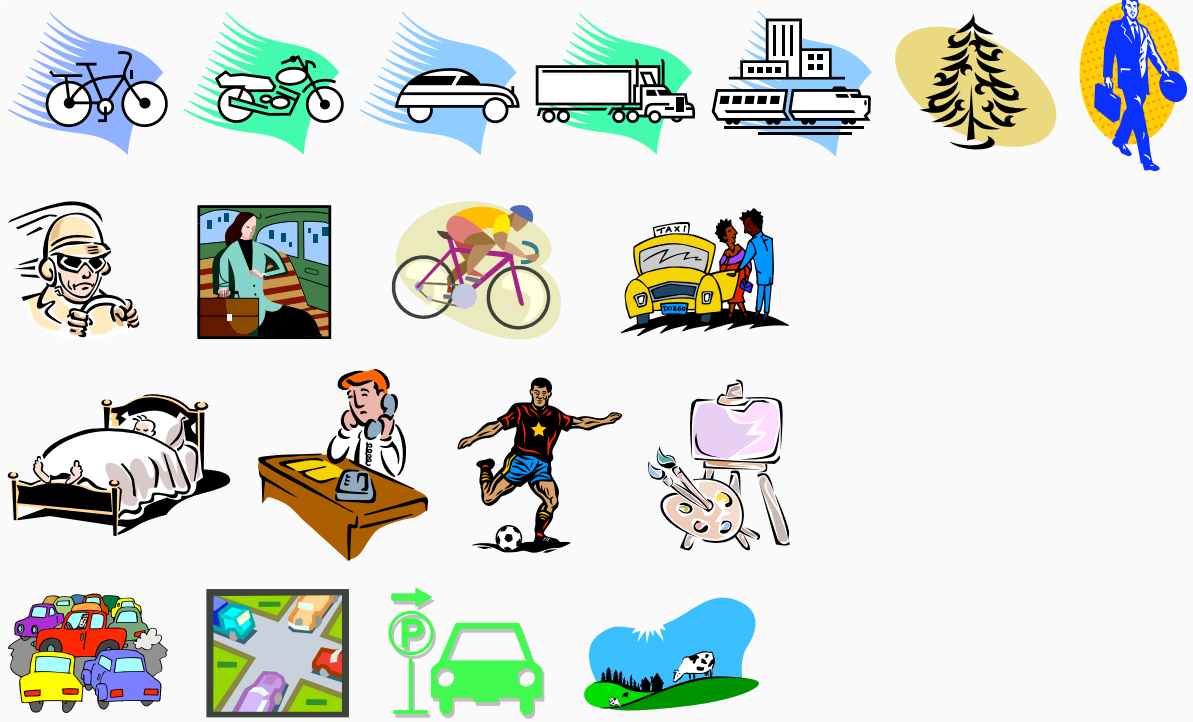
## Cycling Accident

What you hit

Your Role

Activity

Location



# Intelligent Forms

Topics **Angina pectoris**

**Descriptors**

**Presence**

**Duration**

**Severity**

**Progress**

**Associated Symptoms**

**Chest pain**

**Further History**

**Examination**

**Cardiovascular**

**Diagnosis**

**Intervention**

**More on Chest pain**

**Descriptors**

**Chest pain**

**Sublocation**

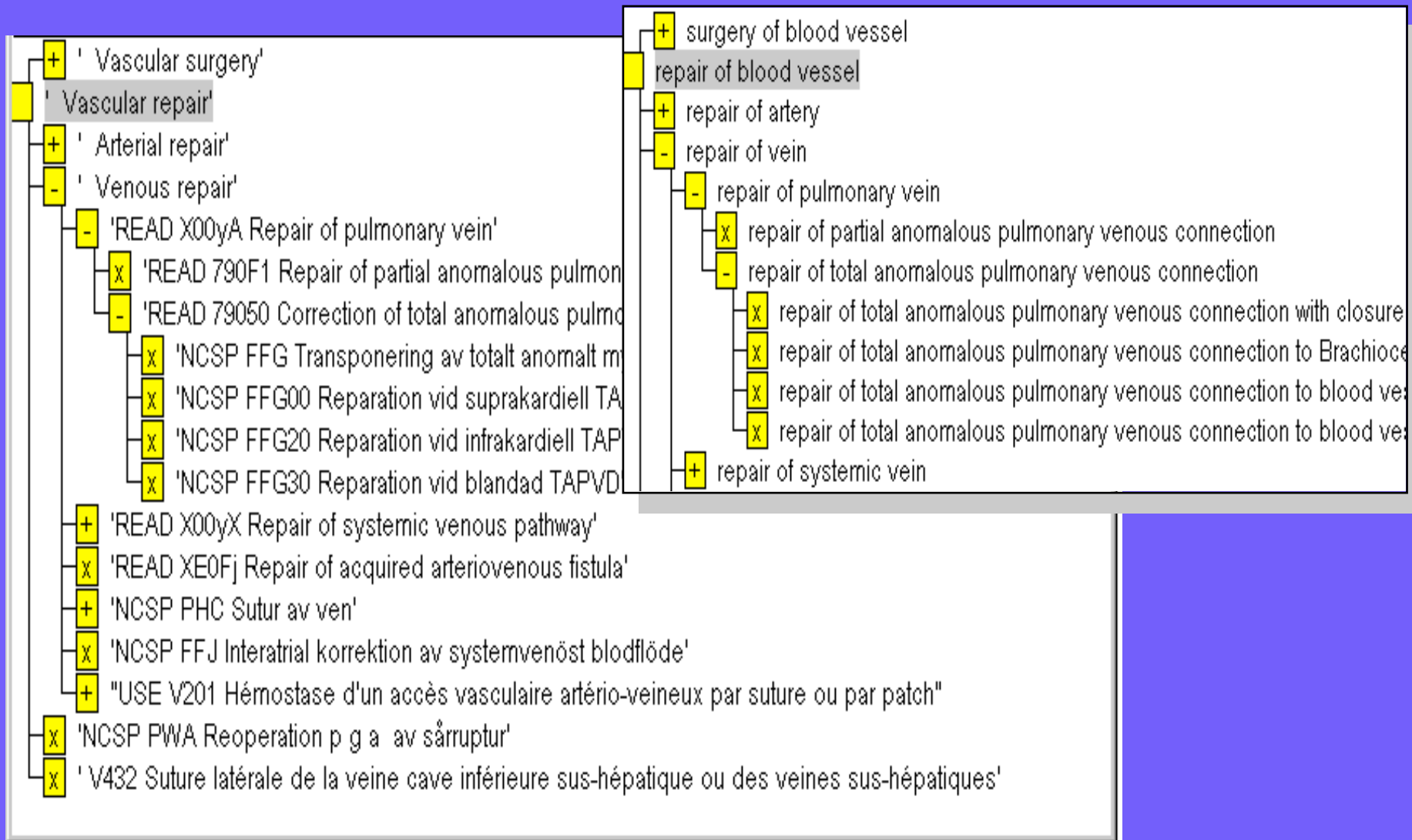
**Duration**

**Character**

**Severity**

**Onset**

# Integrating rather than Cross Mapping





# And generate it in language

## Summary

Moderately severe angina pectoris for 1 day, getting worse

Rapid onset, moderately severe, pressing pain in left chest and sternal region present

## On Examination

Cardiovascular system -

Slightly raised JVP

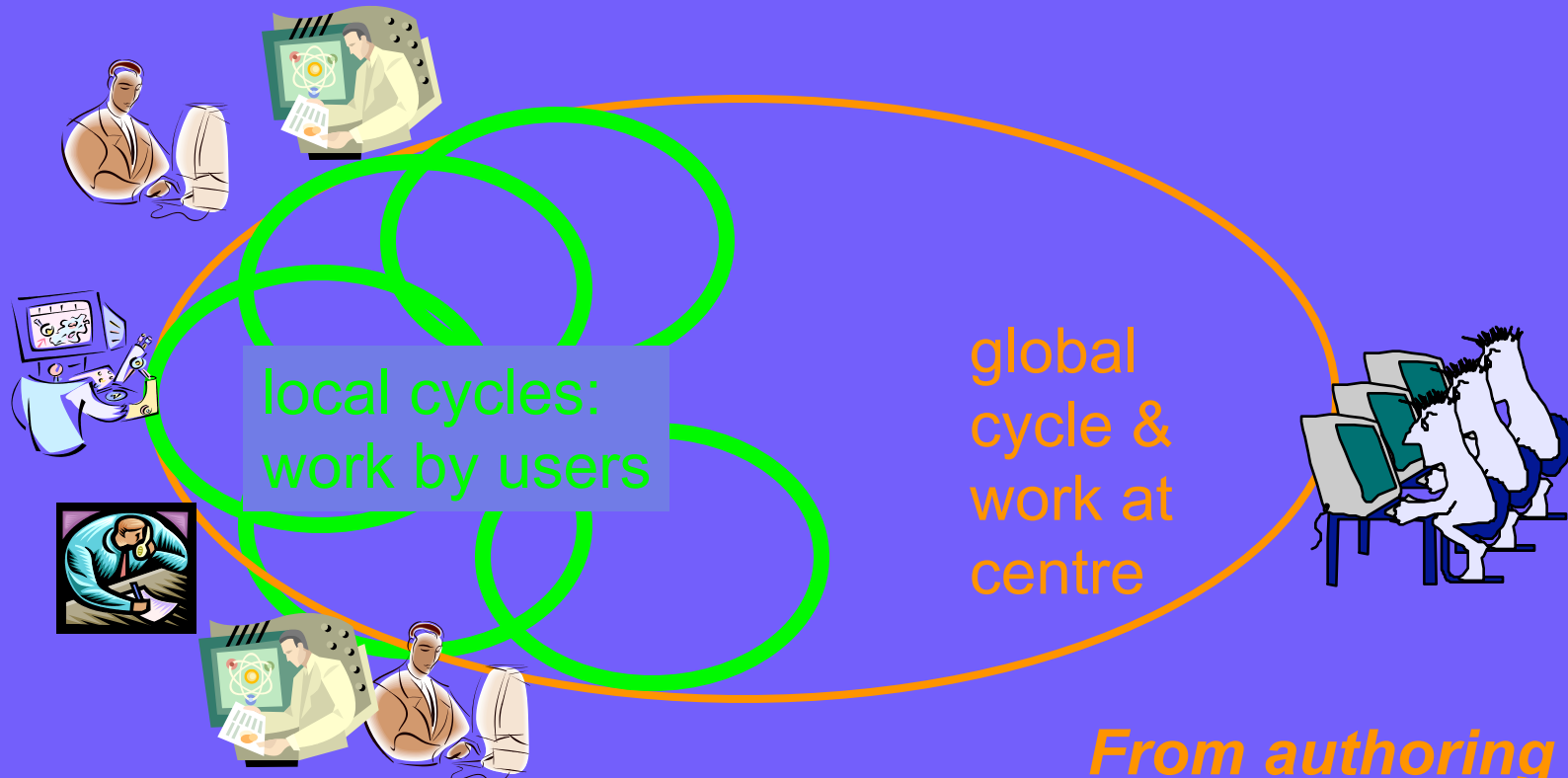
1st and 2nd heart sounds normal

No added heart sounds

Pulse rate 104 per minute

Blood pressure 138/90 mm Hg

# Supports Loosely coupled distributed ontology development



*From 80% central/global effort  
to 10% central/global effort*

*From authoring  
to meta-authoring*

*User effort cut by 75% compared with manual methods  
Mostly in reduced committee meetings & arguments*

# The means: Logic as the clips for “Conceptual Lego”

hand

extremity

body

chronic

acute

abnormal

normal

ischaemic



deletion

polymorphism

mucus

gene

protein

polysaccharide

cell

expression

Lung

infection

inflammation

bacterium

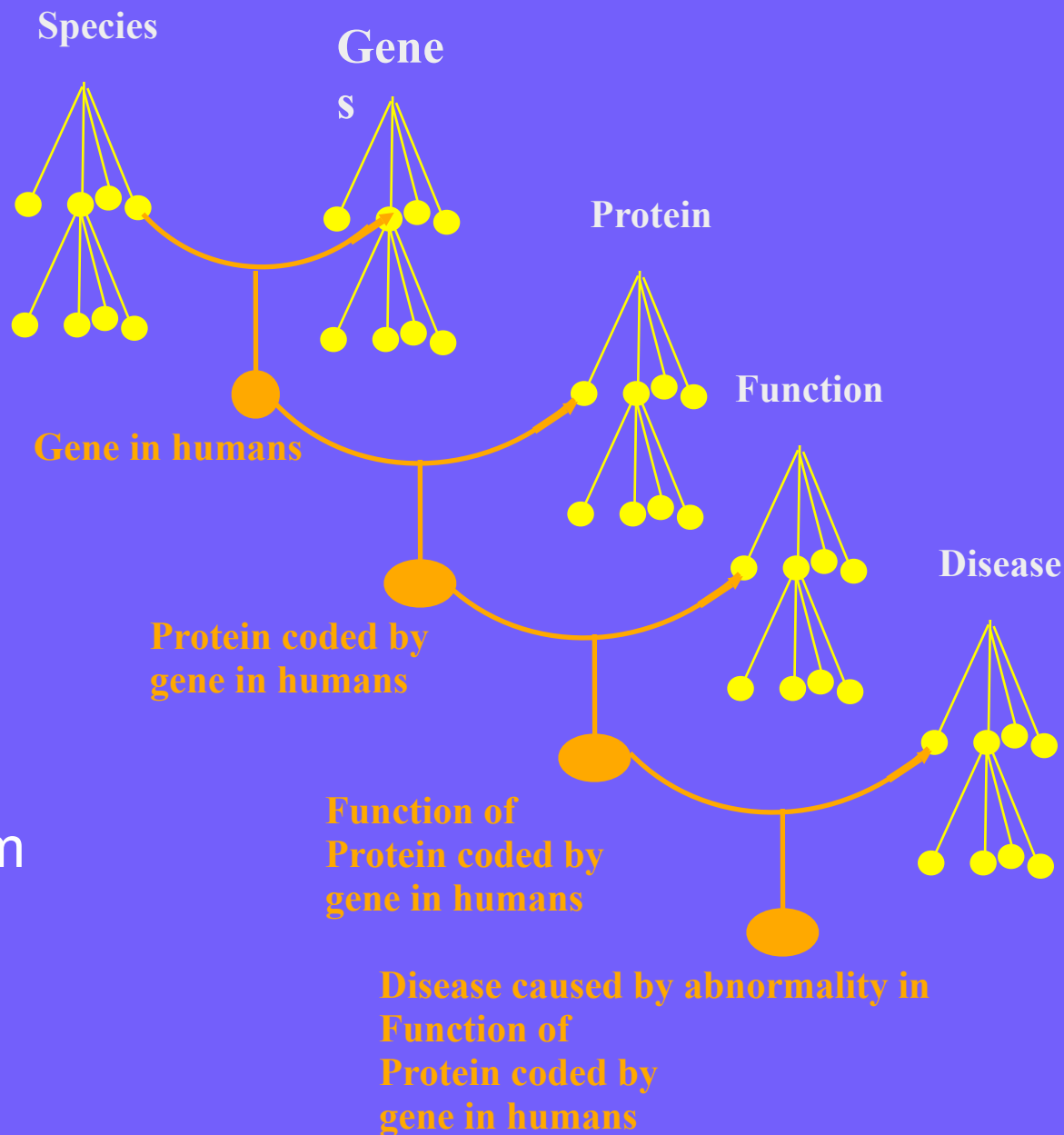
virus

# Logic as the clips for “Conceptual Lego”

“*SNPolymorphism* of *CFTRGene* causing *Defect in MembraneTransport* of *Chloride Ion* causing *Increase* in *Viscosity* of *Mucus* in *CysticFibrosis*...”



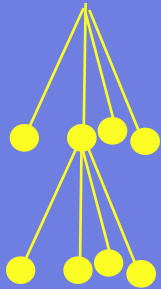
“*Hand* which is *anatomically normal*”



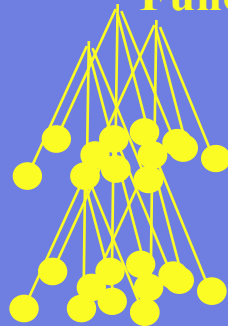
Build complex representations from modularised

# Normalising (untangling) Ontologies

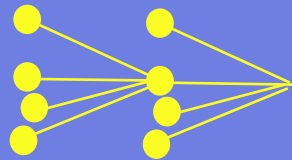
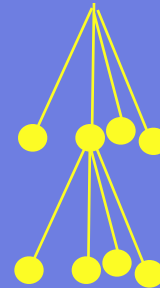
Structure



Structure  
Part-whole  
Function



Function



Part-whole

# Rationale for Normalisation

## ▶ Maintenance

- ▶ Each change in exactly one place
  - ▶ No “Side effects”

## ▶ Modularisation

- ▶ Each primitive must belong to exactly one module
  - ▶ *If a primitive belongs to two modules, they are not modular.*
  - ▶ *If a primitive belongs to two modules, it probably conflates two notions*
- ▶ Therefore concentrate on the “*primitive skeleton*” of the domain ontology

## ▶ Parsimony

- ▶ Requires fewer axioms



# Normalisation and Untangling

Let the reasoner do multiple classification

- ▶ Tree
  - ▶ Everything has just one parent
    - ▶ A 'strict hierarchy'
- ▶ Directed Acyclic Graph (DAG)
  - ▶ Things can have multiple parents
    - ▶ A 'Polyhierarchy'
- ▶ Normalisation
  - ▶ Separate primitives into disjoint trees
  - ▶ Link the trees with restrictions
    - ▶ Fill in the values

# Untangling and Enrichment

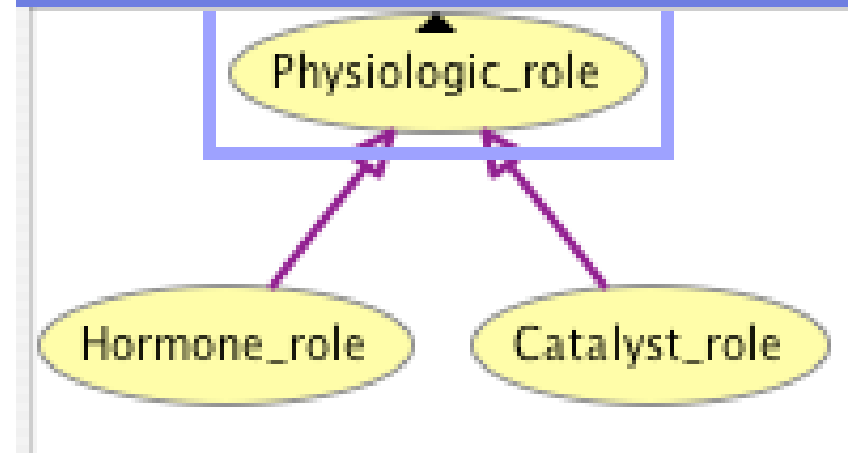
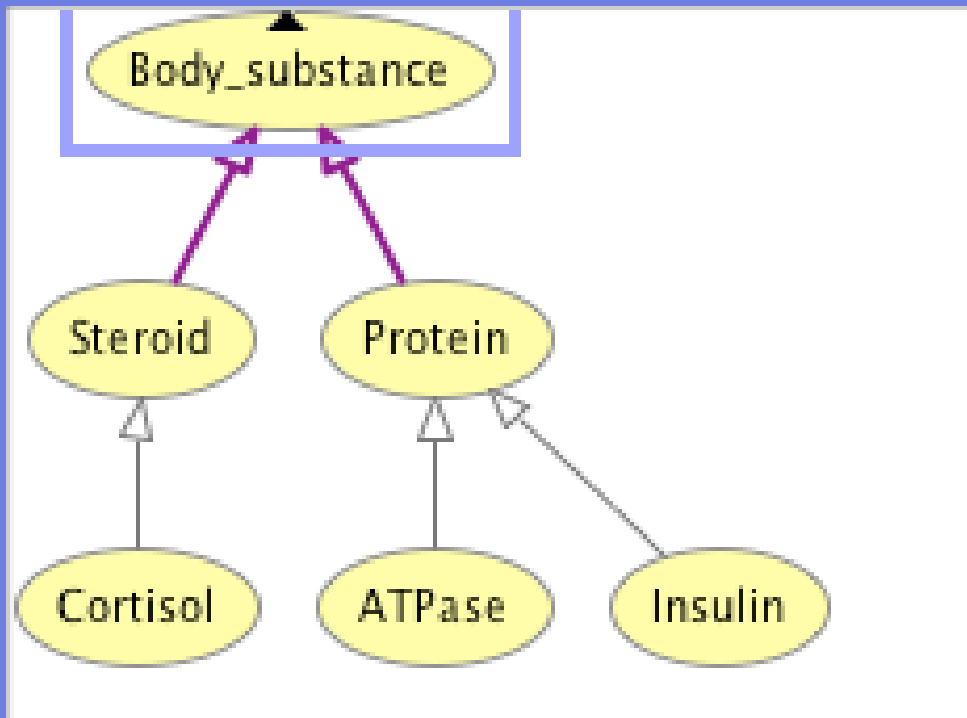
## Using a classifier to make life easier

- **Substance**
- - **Protein**
- - - **Insulin**
- - - **ATPase**
- **Steroid**
- - **Cortisol**

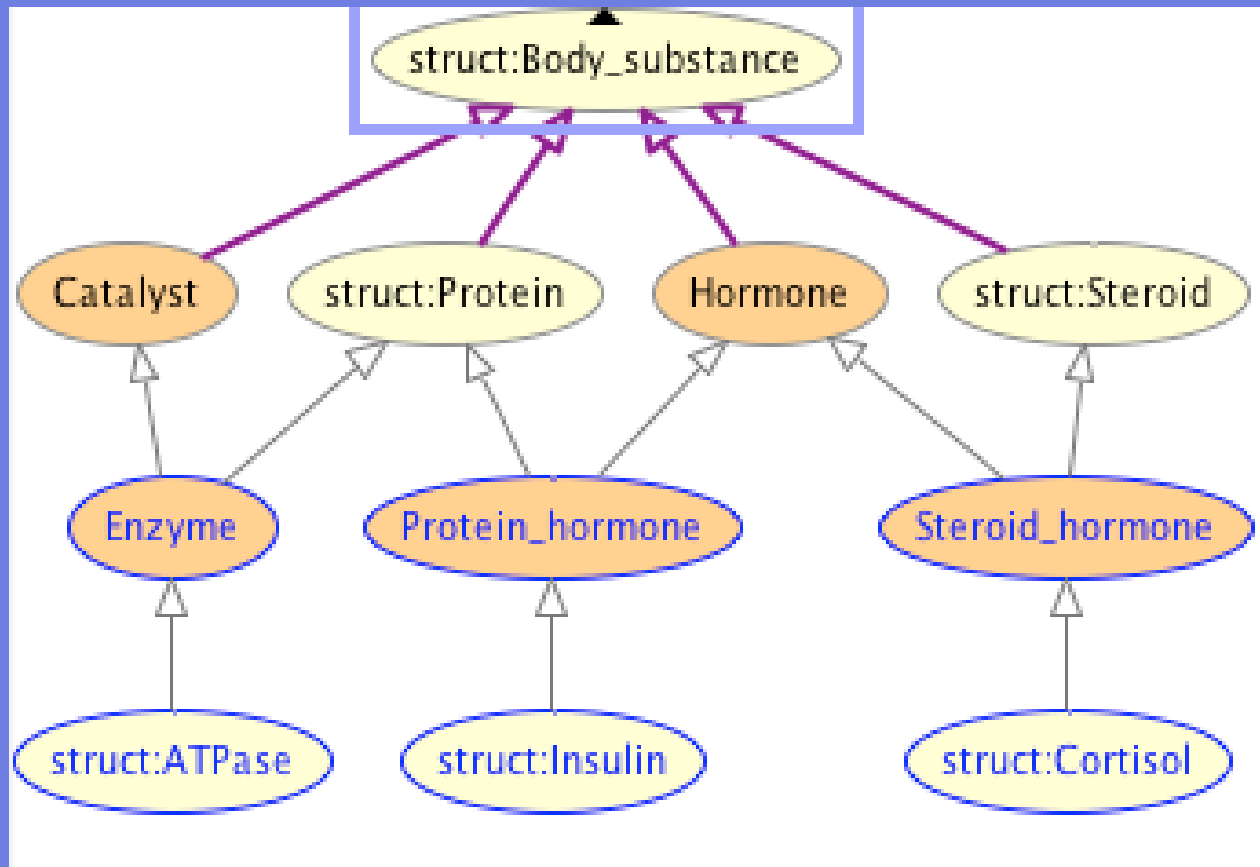
- **PhsioloicRole**
- - **HormoneRole**
- - **CatalystRole**

- Substance**
- **Protein**
- - **ProteinHormone**
- - - **Insulin**
- - **Enzyme**
- - - **ATPase**
- **Steroid**
- - **SteroidHomone^**
- - - **Cortisol**
- **Hormone**
- - **ProteinHormone^**
- - - **Insulin^**
- - **SteroidHormone^**
- - - **Cortisol^**
- **Catalyst**
- - **Enzyme^**
- - - **ATPase^**

# Modularised into structure and function ontologies

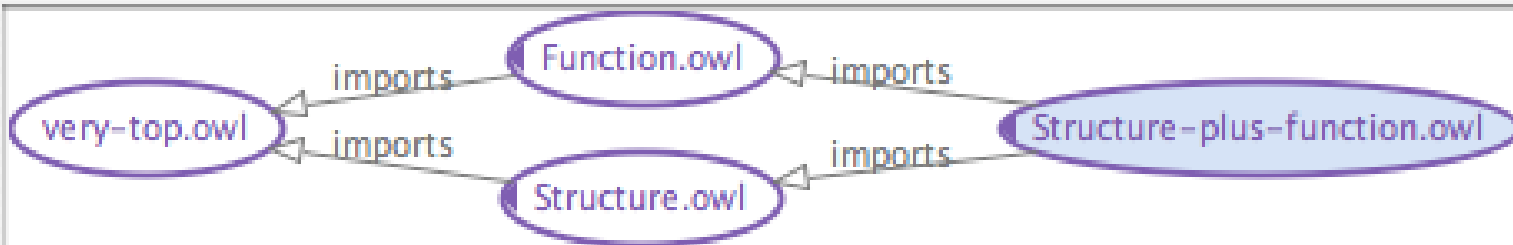


# Unified ontology after classification



# Module Structure

OWLviz Imports Graph:



**NB: Examples are in the tutorial ontology folder**

# Normalisation: Criterion 1

## The skeleton should consist of disjoint trees

- ▶ *Every primitive concept should have exactly one primitive parent*
  - ▶ All multiple hierarchies the result of inference by reasoner

# Normalisation Criterion 2: *No hidden changes of*

▶ *Each branch should be homogeneous and logical*

*(“Aristotelian”)*

- ▶ Hierarchical principle should be subsumption
  - ▶ Otherwise we are “lying to the logic”
- ▶ The criteria for differentiation should follow consistent principles in each branch  
eg. structure *XOR* function *XOR* cause



# Normalisation Criterion 3

## *Distinguish “Self-standing” and “Refining” Concepts*

### *“Qualities” vs Everything else*

- ▶ Self-standing concepts
  - ▶ Roughly Welty & Guarino’s “sortals”
    - ▶ *person, idea, plant, committee, belief,...*
- ▶ Refining concepts – depend on self-standing concepts
  - ▶ *mild|moderate|severe, hot|cold, left|right,...*
  - ▶ Roughly Welty & Guarino’s non-sortals
  - ▶ Closely related to Smith’s “fiat partitions”
  - ▶ Usefully thought of as Value Types by engineers
- ▶ For us an engineering distinction...

# Normalisation Criterion 3a

*Self-standing primitives should be globally disjoint & open*

- ▶ Primitives are atomic
  - ▶ If primitives overlap, the overlap conceals implicit information
- ▶ A list of self-standing primitives can never be guaranteed complete
  - ▶ How many kinds of person? of plant? of committee? of belief?
  - ▶ Can't infer:  $\text{Parent} \ \& \ \neg\text{sub}_1 \ \& \ \dots \ \& \ \neg\text{sub}_{n-1} \ \rightarrow \ \text{sub}_n$
- ▶ Heuristic:
  - ▶ Diagnosis by exclusion about self-standing concepts should *NOT* be part of 'standard' ontological reasoning

# Normalisation Criterion 3b

*Refining primitives should be locally disjoint & closed*

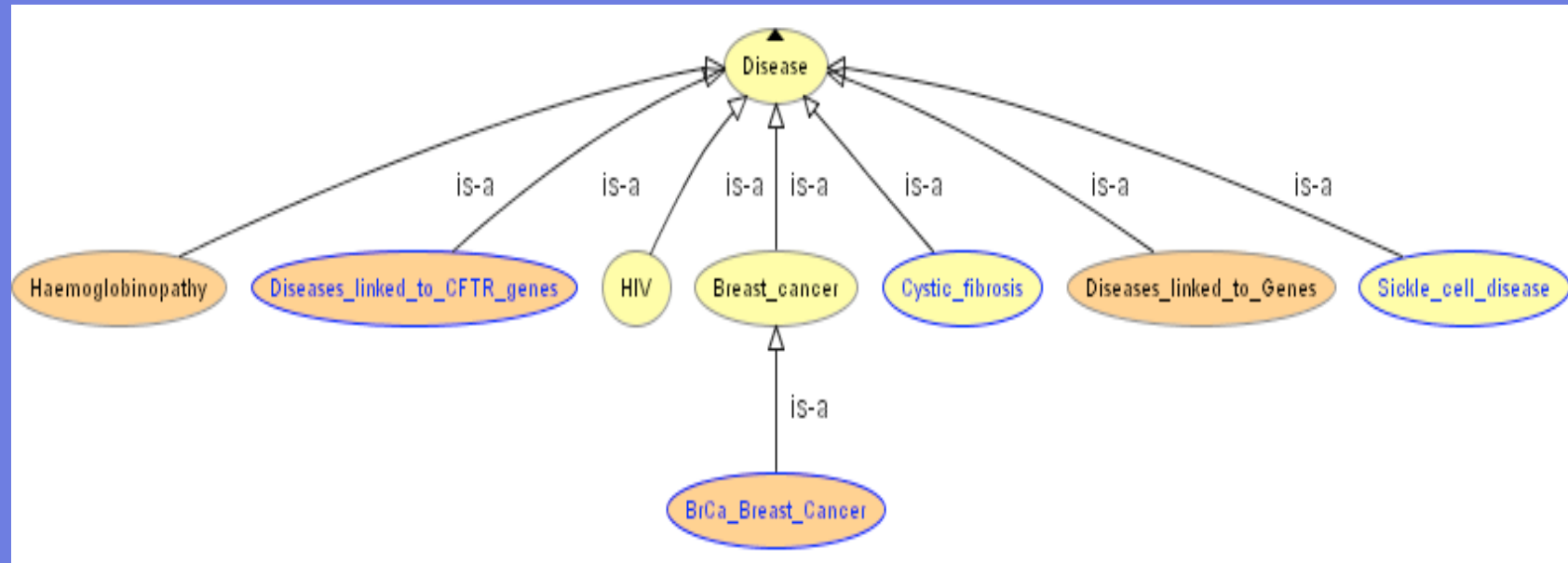
- ▶ individual values must be disjoint
  - ▶ but can be hierarchical
    - ▶ e.g. “very hot”, “moderately severe”
- ▶ Each list can be guaranteed to be complete
  - ▶ Can infer  $\text{Parent} \ \& \ \neg\text{sub}_1 \ \& \ \dots \ \& \ \neg\text{sub}_{n-1} \ \rightarrow \ \text{sub}_n$
- ▶ Value types themselves need not be disjoint
  - ▶ “being hot” is not disjoint from “being severe”
    - ▶ Allowing Valuetypes to overlap is a useful trick, e.g.
      - ▶ restriction has\_state someValuesFrom (severe and hot)

# Normalisation Criterion 4

## Axioms

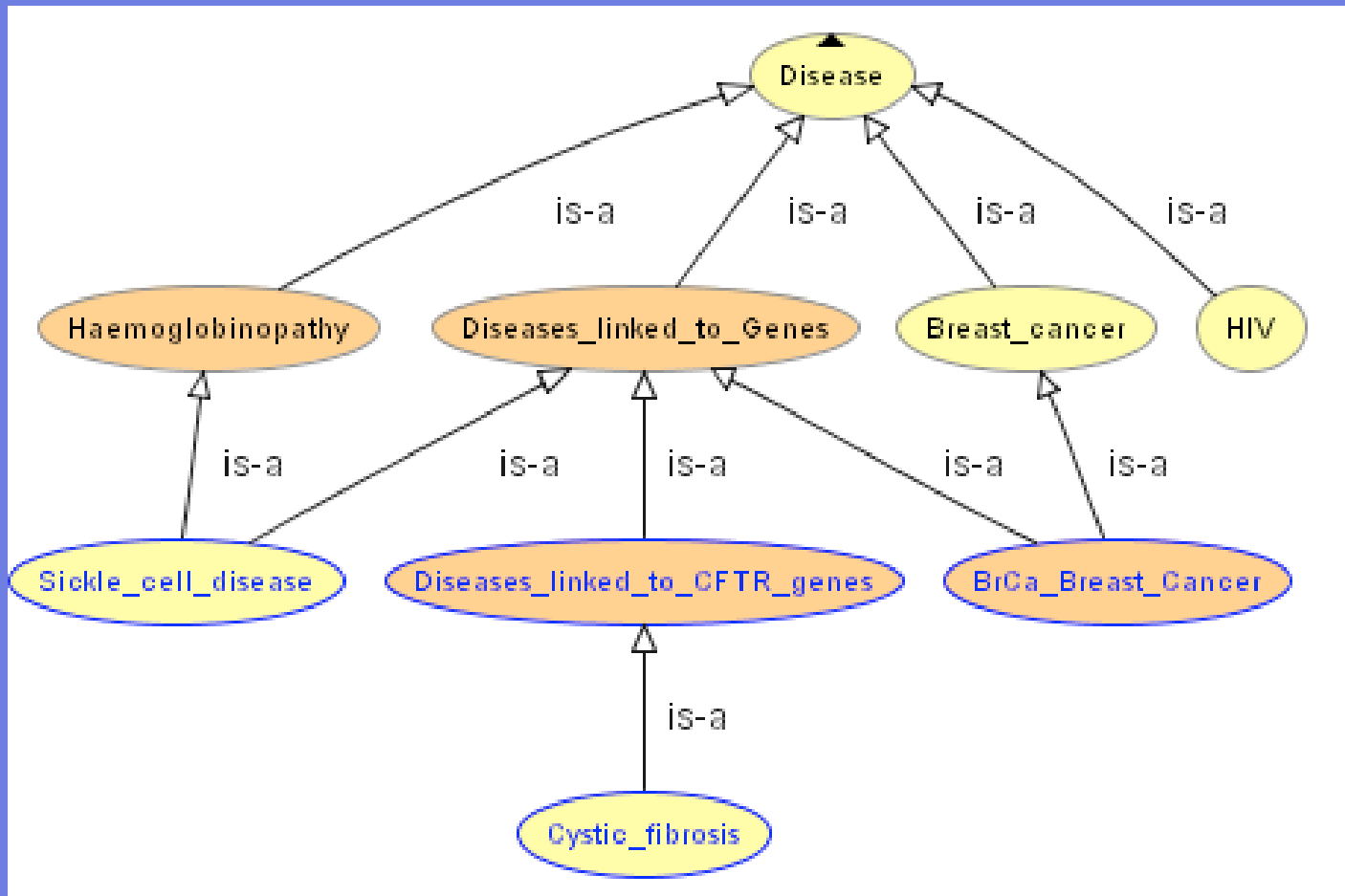
- ▶ *No axiom should denormalise the ontology*  
*No axiom should imply that a primitive is part of more than one branch of primitive skeleton*
  - ▶ **If all primitives are disjoint, any such axioms will make that primitive unsatisfiable**
    - ▶ A partial test for normalisation:
      - ▶ *Create random conjunctions of primitives which do not subsume each other.*
      - ▶ *If any are satisfiable, the ontology is not normalised*

# A real example: Build a simple tree

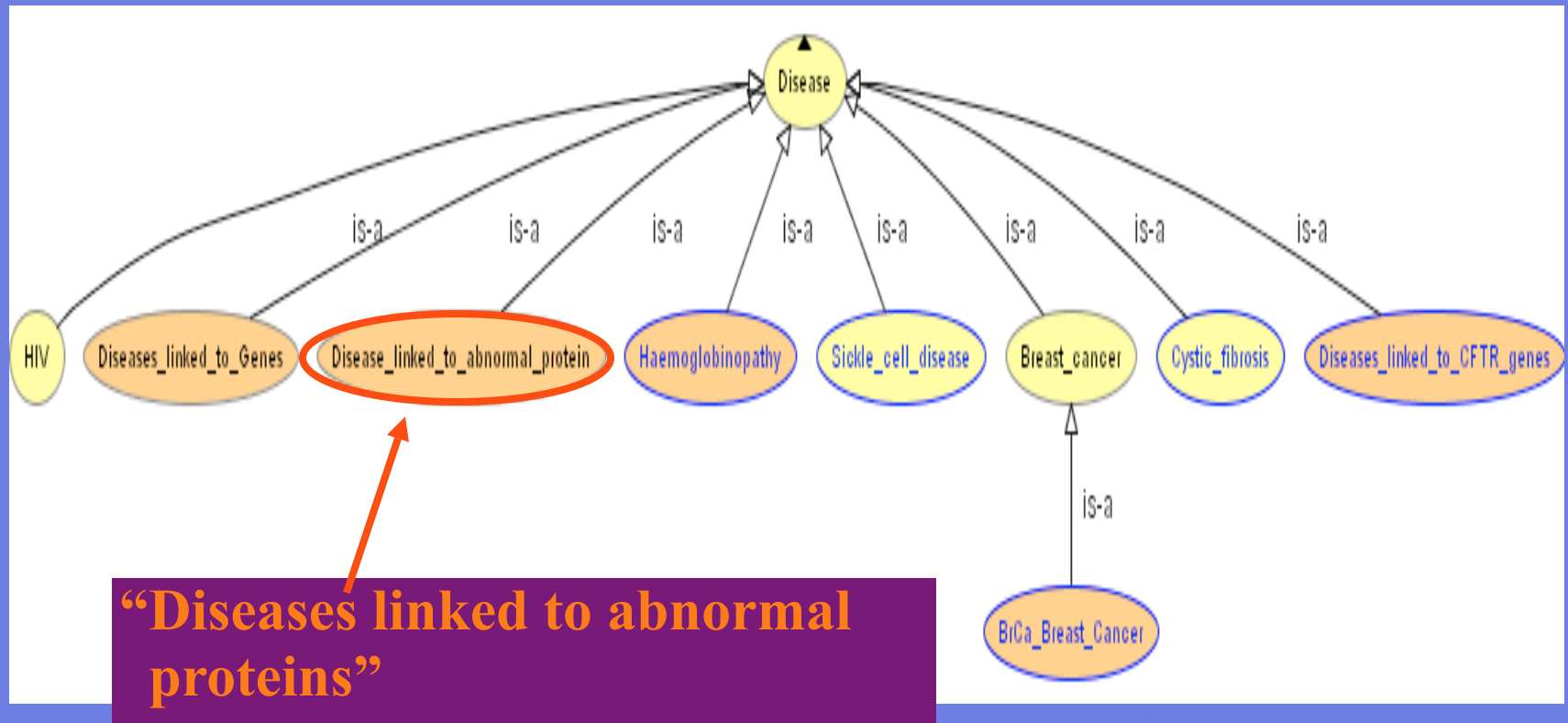


easy to maintain

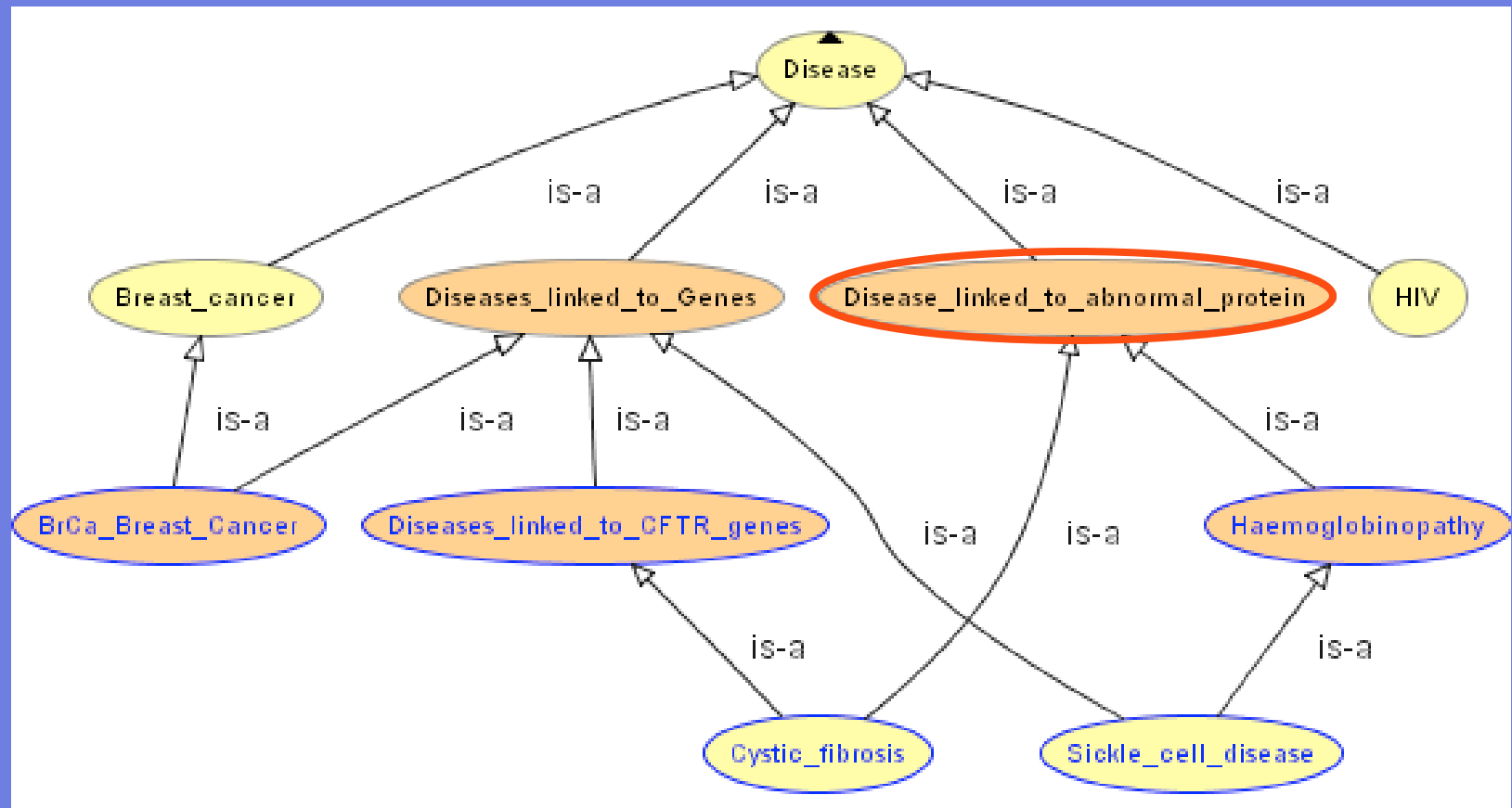
# Let the classifier organise it



# If you want more abstractions,

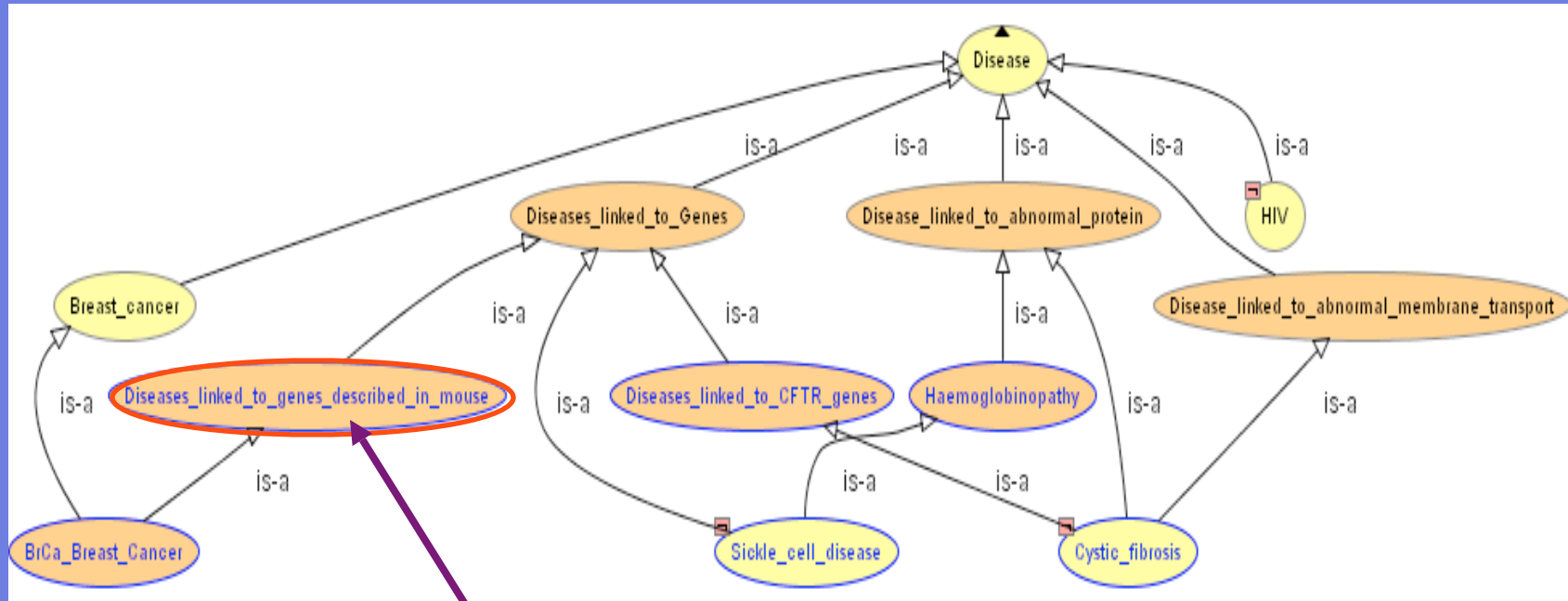


# And let the classifier work again





# And again – even for a quite different category



**“Diseases linked genes described in the mouse”**

# Summary: Why Normalise? Why use a Classifier?

- ▶ To *compose* concepts
    - ▶ *Allow conceptual lego*
  - ▶ To manage *polyhierarchies*
    - ▶ Adding abstractions (“axes”) as needed
    - ▶ *Normalisation*
      - ▶ Untangling
        - ▶ *labelling of “kinds of is-a”*
- ▶ To avoid *combinatorial explosions*
  - ▶ *Keep bicycles from exploding*
- ▶ To manage *context*
  - ▶ *Cross species, Cross disciplines, Cross studies*
- ▶ To check *consistency* and *help users find errors*

# Now: How to do it in OWL - *A quick review*

- ▶ Existential qualifiers (SOME)
- ▶ Universal qualifiers (ONLY)
- ▶ Open World Assumption
  - ▶ Negation as inconsistency ("unsatisfiability")
    - ▶ What is neither provable nor provably false is unspecified ("unknown")
- ▶ Load infections-only.owl
  - ▶ Trivially simple model for illustration only
  - ▶ Define a bacterial infection
  - ▶ Define a viral infection
  - ▶ Define a mixed (bacterial-viral) infection

# Basic OWL

## (In Manchester Syntax)

- ▶ KINDS: B subclassOf A
  - ▶ B --> A
    - ▶ All Bs are As
      - ▶ *All pneumococci are bacteria*
      - ▶ *without exception.*
- ▶ SOME (Existential / someValuesFrom)  
All As have property P with value from V
  - ▶ A --> have\_p SOME V
    - ▶ All As have\_p some kind of V
      - ▶ *Infection --> has\_cause SOME Micro\_organism*
- ▶ ONLY (Universal, allValuesFrom)
  - ▶ A --> have\_p ONLY V
    - ▶ All as have\_p only kinds of V
      - ▶ *Infection --> has\_cause ONLY Micro\_organism*

# Primitive and Defined classes

## ▶ Primitives

### ▶ Named things that are described

▶ Lung -->  
is\_contained\_in SOME Chest

▶ *but so are lots of other things*

▶ *Often related to “natural kinds”*

## ▶ Defined classes

### ▶ Things for which we have a sufficient (and necessary) definition

▶ Pneumococcal\_pneumonia <-->  
Pneumonia AND  
is\_caused\_by SOME Pneumococcus

▶ *“ANYTHING that is a pneumonia and is caused by pneumococcus is a ‘pneumococcal pneumonia’*

# Domain and Range constraints

- ▶ Domain and range constraints are really disguised universals
- ▶ *has\_part* DOMAIN Anatomical\_structure  
RANGE Anatomical\_structure  
means
- ▶ Thing *has\_part* ONLY Anatomical\_structure  
Thing *is\_part\_of* ONLY Anatomical\_structure
- ▶ BEWARE: Domain and Range constraints can effect classification -
  - ▶ A common source of errors

# Lack of Unique Name Assumption

- ▶ In OWL, anything may be the same until we say it is disjoint or different
- ▶ For classes
  - ▶ Micro-organism
    - Bacterium
    - Virus
  - ▶ DISJOINT( Bacteria, Virus)
    - ▶ Otherwise may overlap
- ▶ For individuals
  - ▶ allDifferent( male, female )

# Open World Reasoning

- ▶ Everything is true unless it can be proved false
  - ▶ False means “probably false”
    - ▶ “Open World Assumption”
      - ▶ *All OWL definitions and descriptions implicitly contain “amongst other things”*
        - ▶ *“Pneumococcal pneumonia is a pneumonia that, amongst other things, is caused by pneumococcus”*
- ▶ Most other systems use “negation as failure”
  - ▶ If you can't find it, it must be false
    - ▶ “Closed World Assumption”



# The OWL Reasoner

- ▶ Organises the subclass ( “subsumption”) hierarchy according to the definitions (and other axioms)
  - ▶ A specialised theorem prover
  - ▶ AKA “classifier”
- ▶ Several available
  - ▶ FaCT++ and Pellet are built into Protege4Alpha
    - ▶ FaCT++ is (usually) faster
    - ▶ Pellet is more robust
    - ▶ Pellet will soon have better debugging facilities

# Basic steps in building an ontology:1

## Gather your terms

Lung

Pneumonitis

Rales

Chest

Pneumonia

Bacteria

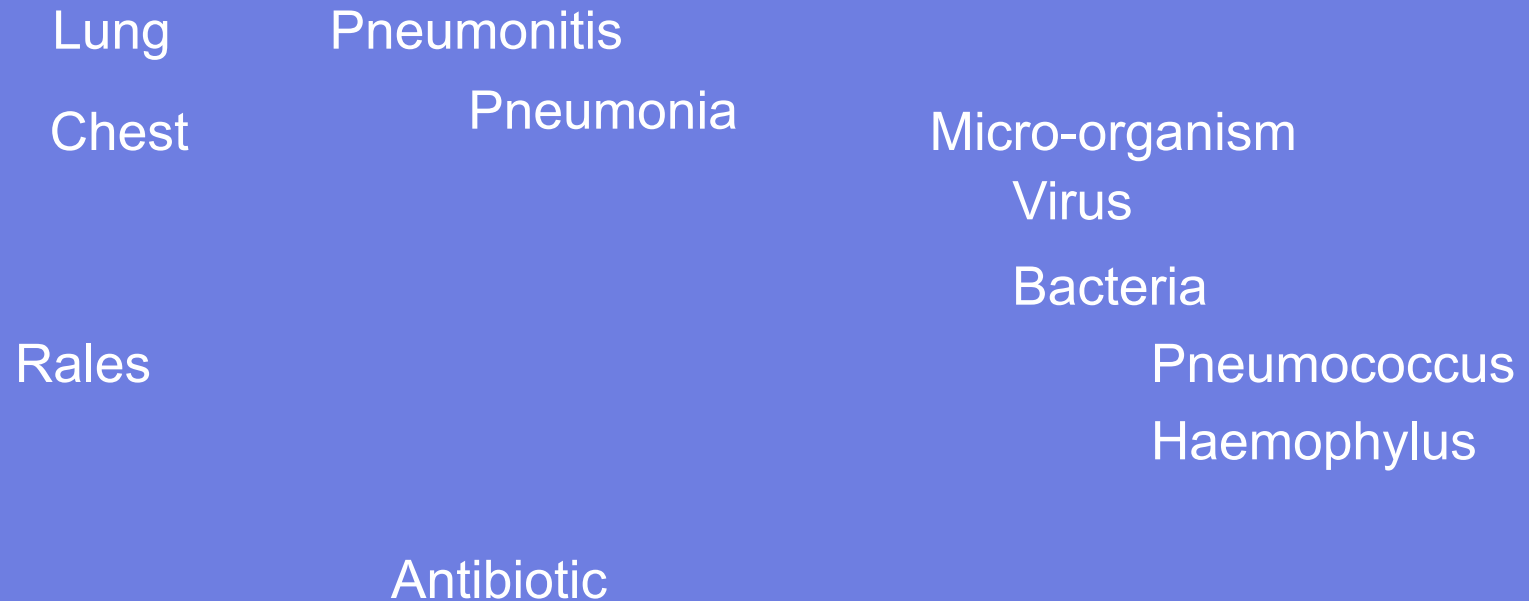
Virus

Antibiotic

Haemophylus

Pneumococcus

# Basic steps in building an ontology 2: Organise your terms



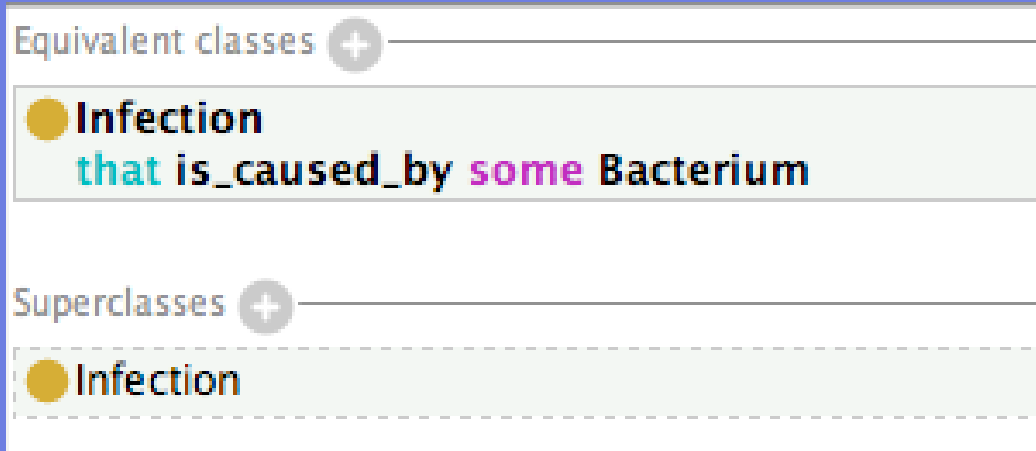
# Basic steps in building an ontology 3

## Represent your terms

- ▶ Perhaps first using mind maps or CMAPS
- ▶ Then when you understand them in OWL
  - ▶ Including definitions
  - ▶ And not throwing away the original source material
- ▶ ... So now on to OWL
  - ▶ First,  
A simplistic version of Pneumonia & Pneumonitis

# Bacterial infection

- ▶ *Any* infection caused by *some* bacterium




- ▶ Note that it is a defined (equivalent) class.

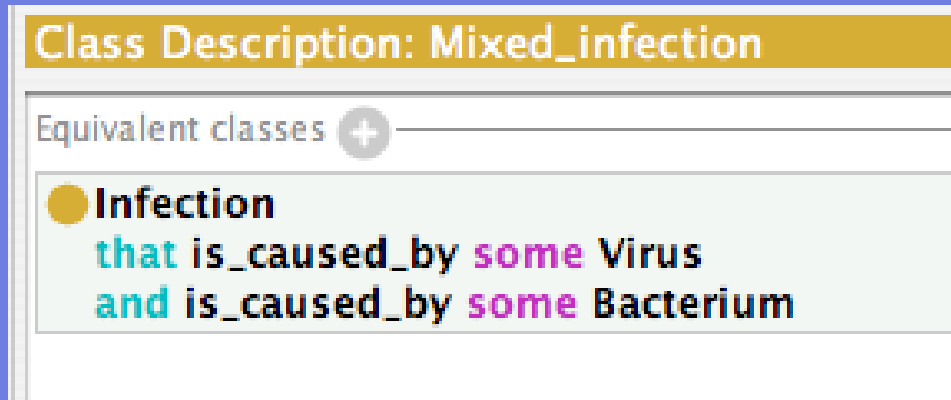
# Viral infection

## Class Description: Viral\_infection

Equivalent classes 

-  **Infection**  
that is\_caused\_by some Virus

# Mixed infection



The screenshot shows a software interface for a class description. At the top, a yellow header bar contains the text "Class Description: Mixed\_infection". Below this, a section titled "Equivalent classes" with a plus sign icon is visible. A radio button next to the word "Infection" is selected. The text below the radio button reads: "that is\_caused\_by some Virus and is\_caused\_by some Bacterium".

- ▶ How will bacterial infection, viral infection, and mixed infection classify?
  - ▶ Run the classifier and see

# After classification

- ▼ ● Domain\_entity
  - ▶ ● Continuant
  - ▼ ● Occurrent
    - ▼ ● Infection
      - ▼ ☰ Bacterial\_infection
        - ☰ Mixed\_infection
      - ▼ ☰ Viral\_infection
        - ☰ Mixed\_infection

Why?



# A “Pure bacterial infection”

## Class Description: Pure\_bacterial\_infection

Equivalent classes 

- **Infection**  
that is\_caused\_by **some** Bacterium  
and is\_caused\_by **only** Bacterium

## ▶ A pneumococcal infection

## Class Description: Pneumococcal\_and\_haemophyllus\_infection

Equivalent classes 

- **Infection**  
that is\_caused\_by **some** Haemophyllus  
and is\_caused\_by **some** Pneumococcus

- ▶ How will these classify
  - ▶ Is a pneumococcal and haemophyllus infection a kind of pure bacterial infection? - Both are

# Why not?

- ▼ **☰ Bacterial\_infection**
  - ☰ Mixed\_infection
  - ☰ **Pneumococcal\_and\_haemophylus\_infection**
  - ☰ Pure\_bacterial\_infection
- ▶ **☰ Viral\_infection**

# Closure Axioms

Equivalent classes 

## ● Infection

that is\_caused\_by some Haemophyllus  
and is\_caused\_by some Pneumococcus  
and is\_caused\_by only (Haemophyllus  
or Pneumococcus)

# Trivial satisfiability

## Two common errors

Class Description: Probe\_Only\_bacterial\_infection

Equivalent classes 

- Infection  
that is\_caused\_by only Infection

Class Description: Probe\_Mixed\_infection\_wrong

Equivalent classes 

- Infection  
that is\_caused\_by only (Bacterium  
and Virus)

▶ How will these classify? Why?

# After classification (Explain it)

Class Description: Probe\_Only\_bacterial\_infection

Equivalent classes +

- Infection  
that is\_caused\_by only Infection

Class Description: Probe\_Mixed\_infection\_wrong

Equivalent classes +

- Infection  
that is\_caused\_by only (Bacterium  
and Virus)

- ▼ ● Infection
  - ▶ ≡ Bacterial\_infection
  - ▼ ≡ Probe\_Only\_bacterial\_infection
    - ≡ Probe\_Mixed\_infection\_wrong
    - ≡ Pure\_bacterial\_infection
  - ▶ ≡ Viral\_infection

# Trivial satisfaction

- ▶ Bacterium and Virus are disjoint
  - ▶ Nothing is both a bacterium and a virus
    - ▶ owl:Nothing = (Bacterium AND Bottom)
- ▶ ONLY NOTHING  $\leftrightarrow$  NOT SOME THING
  - ▶ Infection THAT is\_caused\_by ONLY Nothing = Infection THAT NOT (is\_caused\_by SOME Thing)
- ▶ ONLY does not mean SOME
  - ▶ Infection THAT is\_caused\_by ONLY Bacterium = Infection THAT NOT (is\_caused\_by SOME NOT Bacterium)
    - ▶ *No cause given. There may be a cause, as long as it is a*
- ▶ Therefore:

An infection not caused by anything is a kind of infection not caused by anything except bacteria.

  - ▶ Check definition for “Pure bacterial infection”

**Modularisation:**  
*towards assembling  
ontologies from reusable  
fragments*

# Why use modules

- ▶ Re-use
  - ▶ e.g. annotations, quantities, upper ontologies
- ▶ Coherent extensions
  - ▶ Localisation & Views
    - ▶ Local normal ranges, value sets, etc. under generic headings
  - ▶ Experimentation and add ins
    - ▶ e.g. add in tutorial examples without corrupting basic structure
- ▶ Logical separation
  - ▶ e.g. avoid confusing medicine and medical records
- ▶ ...but managing modularised ontologies is more work
  - ▶ More things to remember.
  - ▶ More things to get wrong



# Modules and imports

## ▶ Key notions:

### ▶ “Base URI” - the identifier for the ontology

▶ In the form of a URI but really just an ID

▶ *Used by the import mechanism to identify the module*

### ▶ Physical location

▶ Where the module is actually stored.

▶ *usually your local directory for this version of the ontology*

## ▶ Our conventions:

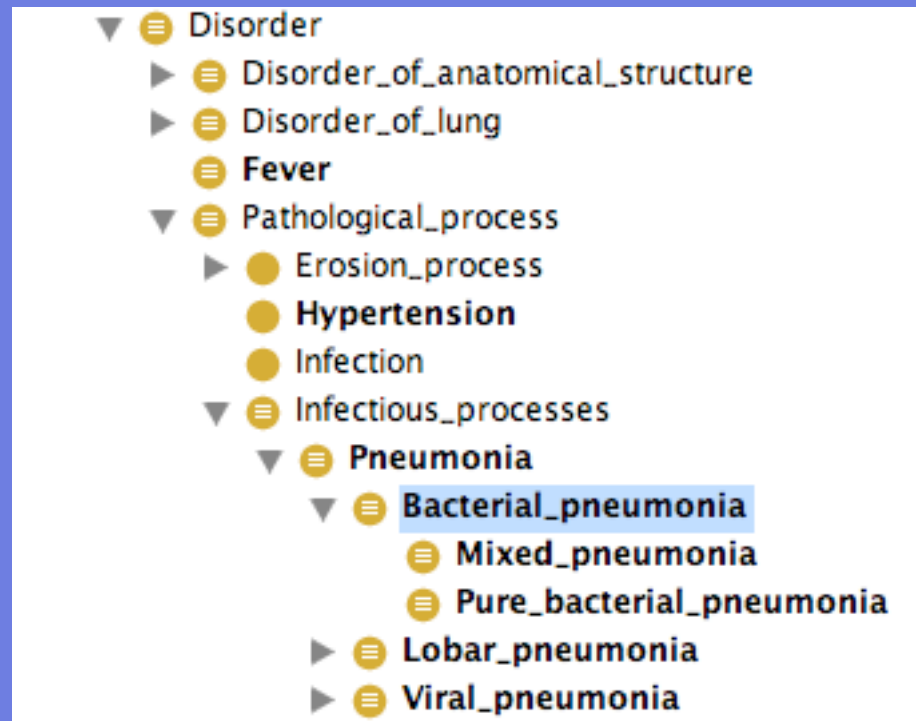
▶ Ontologies stored as sets of modules in a single directory

▶ “Start-Here.owl” tells you what to load and load everything else.

▶ The “Active ontology” is the one you are editing

▶ Active ontology items are shown in bold

# Items from active ontology are in bold



# Protege-OWL import mechanism

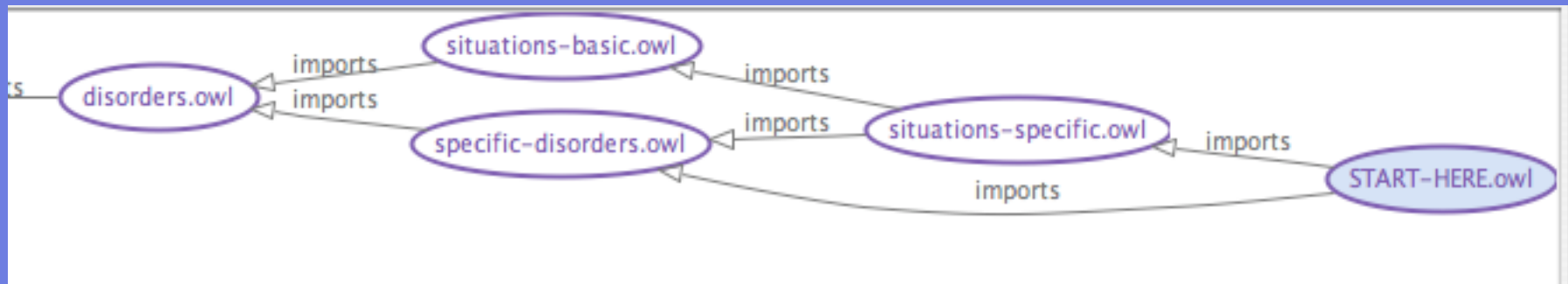
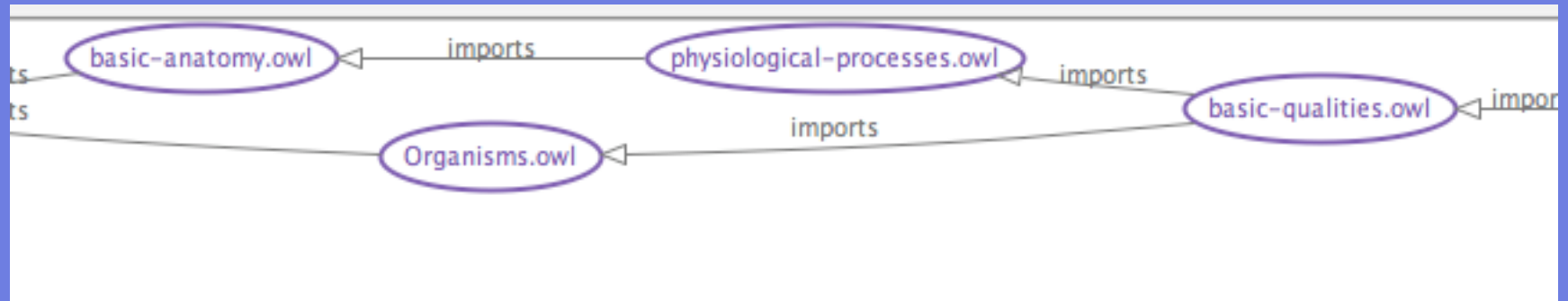
- ▶ Importer looks for a file with the correct identifying “Base URI”
  - ▶ Written into the header of the XML
  - ▶ NOT the physical location
- ▶ Order of search
  - ▶ (Specified file)
  - ▶ The local directory
  - ▶ Local libraries
  - ▶ Global libraries
  - ▶ The internet at the site indicated by the Base URI

# If you can, load the tutorial ontology nowontology/

- ▶ Open  
.../biomedical-tutorial-2007/Start-Here.owl

# Typical pattern

Views→Ontology views→OwlViz Imports



# Module list

- ▶ Annotation - the annotation properties needed
- ▶ generic-data-structures -
  - ▶ quantities & numbers
- ▶ very-top - the upper ontology
  - ▶ in this case adapted for biomedicine
- ▶ Anatomy, Physiology, Biochemistry, Organism
  - ▶ The main topics
- ▶ Qualities
  - ▶ The basic qualities of those topics
- ▶ Disorders
  - ▶ General patterns for disorders
- ▶ Specific disorders
  - ▶ Examples for this tutorial
- ▶ Situations
  - ▶ Disorders in context of patients and observations

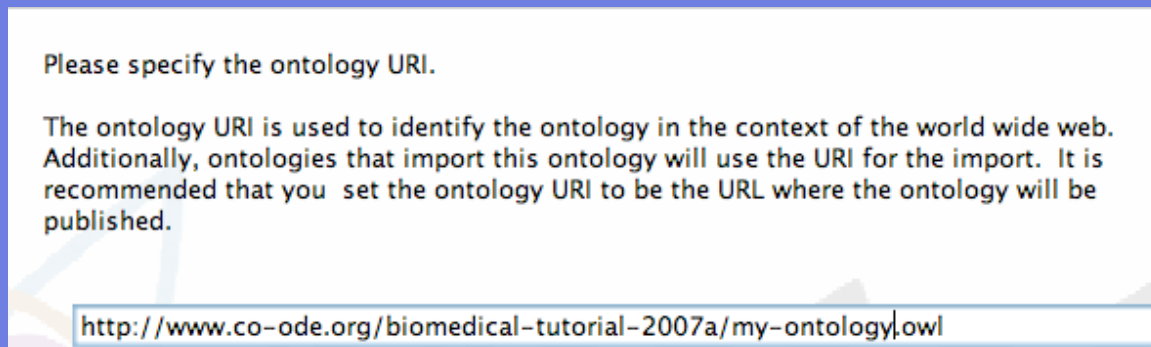
# Anatomy and Disorders

## A more realistic version of Pneumonias

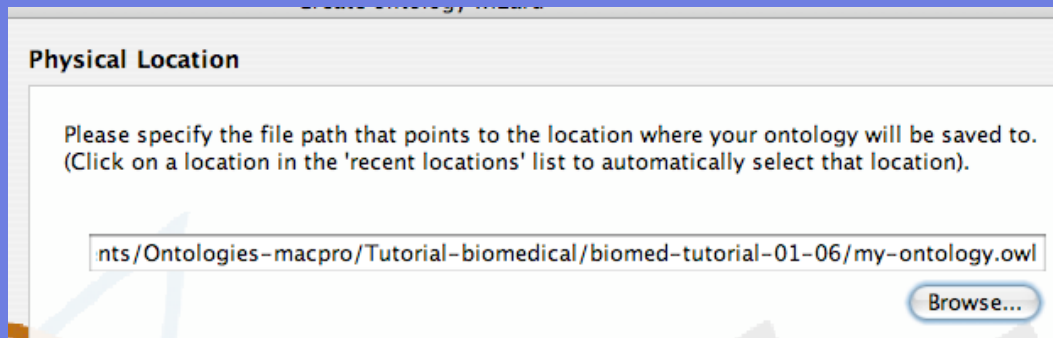
- ▶ Disorders have a locus in an anatomical structure of physiological process
  - ▶ Disorder has\_locus SOME Anatomical\_structure
- ▶ Disorders are anything which is described as pathological
  - ▶ has\_normality\_quality SOME Pathological
    - ▶ To be explained in detail late
- ▶ Parts and wholes
  - ▶ A whole field “mereology”
  - ▶ Multiple views - functional / clinician’s view different from structural / anatomist’s view.

# Examples

- ▶ Backup your ontology directory
- ▶ Make “disorders.owl” the active ontology
- ▶ Create a new ontology in the same frame named “my-disorders”
  - ▶ File new
  - ▶ When pop-up asks about a new frame say NO
  - ▶ When asked for a name edit the end of the URI to “my-ontology.owl”



- ▶ When asked where to store it, browse to your current directory

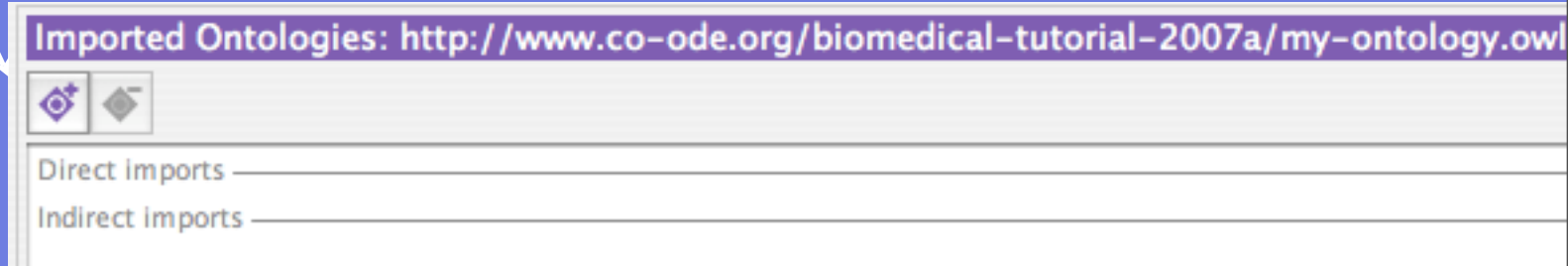


- ▶ Press finish
- ▶ NB This does NOT save your ontology!



# Import disorders.owl

- ▶ Go to the Active Ontology Tab
- ▶ Click the plus icon for imported ontologies



- ▶ Select import an ontology that has already been loaded

- Import an ontology that is contained in one of the ontology libraries.
- Import an ontology that has already been loaded/created.
- Import an ontology contained in a document located on the web.
- Import an ontology contained in a specific file.

- ▶ Select disorders.owl and press finish

# Task: Make Pneumonitis and Pneumonias in various variations

## ▶ Question 1: What is “Pneumonia” and what is “Pneumonitis”

### ▶ Look it up

▶ e.g. Google define: pneumonitis

#### Definitions of **pneumonitis** on the Web:

- inflammation of the lungs; caused by a virus or an allergic reaction  
[wordnet.princeton.edu/perl/webwn](http://wordnet.princeton.edu/perl/webwn)

## ▶ Write your own paraphrases:

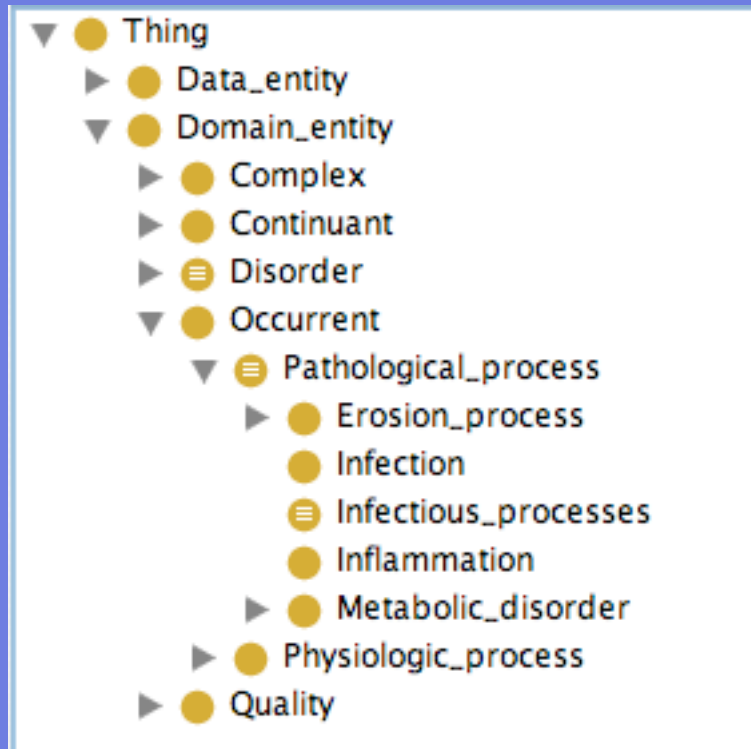
▶ “Pneumonitis” is an “Inflammation of the lungs”

▶ “Pneumonia” is an “inflammation of the lungs caused by an infection”

▶ Many definitions on the web, but this summarises them for our purposes.

# First definition of pneumonitis

- ▶ “Inflammation of the lung”
  - ▶ Find Inflammation
    - ▶ CTRL or CMND F in class hierarchy




# Create “Pneumonitis”


- ▶ Create a new subclass of Inflammation
  - ▶ In the comment box type something like “Pneumonitis” = “Inflammation of lung”
    - ▶ ALWAYS add a free text paraphrase of what you are modelling
  - ▶ Add the restriction
    - ▶ has\_locus SOME Lung
  - ▶ Make it a defined class
    - ▶ CTRL/CMD-D.

Property	Value
comment	Pneumonitis is an inflammation of the Lungs

**Class Description: Pneumonitis**



Equivalent Class (Necessary & Sufficient Criteria) \_\_\_\_\_

 **Inflammation**  
**has\_locus some Lung**

Subclass Of (Necessary Criteria) \_\_\_\_\_




Inferred/Inherited anonymous descriptions (Necessary criteria) \_\_\_\_\_

# Create pneumonia

- ▶ “Pneumonia” is a pneumonitis is the outcome of an infection
  - ▶ In this ontology we use “is\_outcome\_of” for “cause”

Property	Value
comment	Pneumonia is a Pneumonitis that is the outcome of an infection.

Class Description: Pneumonia	
	
Equivalent Class (Necessary & Sufficient Criteria)	_____
 <b>Pneumonitis</b>	
 <b>is_outcome_of some Infection</b>	
Subclass Of (Necessary Criteria)	_____
Inferred/Inherited anonymous descriptions (Necessary criteria)	_____

# Bacterial pneumonia

- ▶ First attempt
  - ▶ “Pneumonia caused by a bacteria”
- ▶ But need to rephrase to fit the ontology
  - ▶ “Pneumonia that is the outcome of an infection by bacteria”
    - ▶ In this ontology “by” translates to the property “has\_actor”
    - ▶ *Processes have actors and objects*

The screenshot displays two panels from an ontology editor. The top panel, titled "Class Annotations: Bacterial\_pneumonia", contains a table with the following data:

Property	Value	Lang
comment	"Bacterial pneumonia is a Pneumonia that is the outcome of an infection by bacteria"	

The bottom panel, titled "Class Description: Bacterial\_pneumonia", shows the class hierarchy and descriptions:

- Equivalent Class (Necessary & Sufficient Criteria) —
- Subclass Of (Necessary Criteria) —
- ☐ Pneumonia
- ☐ is\_outcome\_of some (Infection that (has\_actor some Bacterium))
- Inferred/Inherited anonymous descriptions (Necessary criteria) —

# By analogy make viral pneumonia and mixed pneumonia

- ▶ Mixed pneumonia is a pneumonia that is caused by both virus and pneumonia

- ▶ How to say this



The screenshot shows a software interface with two main sections:

- Class Annotations: Mixed\_pneumonia**: A table with two columns: 'Property' and 'Value'. The 'comment' property has the value: "Mixed pneumonia" is a pneumonia caused by both virus and bacteria.
- Class Description: Mixed\_pneumonia**: A section with several icons at the top. Below, it lists criteria for 'Pneumonia':
  - Equivalent Class (Necessary & Sufficient Criteria):
    - Pneumonia**
    - is\_outcome\_of some (Infection that (has\_actor some Bacterium))
    - is\_outcome\_of some (Infection that (has\_actor some Virus))
  - Subclass Of (Necessary Criteria):
  - Inferred/Inherited anonymous descriptions (Necessary criteria):

- ▶ **WARNING**

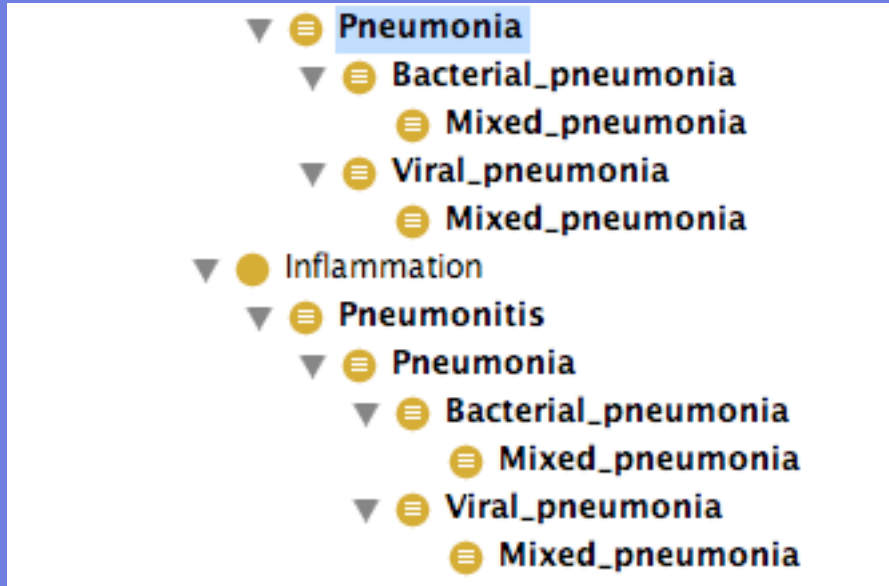
- ▶ wrong: has\_actor SOME (Virus AND Bacterium)
    - ▶ *Nothing is both a virus and a bacteria*

# Classify and check

- ▶ Be sure that all classes are defined
  - ▶  defined
  - ▶  primitive
- ▶ To convert from primitive to defined, cmnd-d or ctrl-d (Mac or PC)



# Should get



# What about “left lower lobe pneumonia”?

- ▶ First define lobar pneumonia as
  - ▶ Pneumonia that has locus in a lobe of a lung
    - ▶ “Lobe THAT is\_subdivision\_of SOME Lung”
- ▶ But what then is a disorder of the lung
  - ▶ Disorder THAT has\_locus SOME Lung
- ▶ But what if I define an inflammation of a lobe of the lung
  - ▶ Inflammation THAT has\_locus SOME (Lobe THAT is\_subdivision\_of SOME Lung)
  - ▶ The classifier ought to organise it for us
    - ▶ ... but it doesn't.

```
▼ ⓘ Disorder_of_anatomical_structure
  ▶ ⓘ Disorder_of_lung
    ▼ ⓘ Inflammation_of_lobe_of_lung
      ⓘ Lobar_pneumonia
```

# OWL means what it says

- ▶ Lobes are not lungs!
  - ▶ Our definition of lung disorder is too narrow
    - ▶ Almost always
      - Disorders of parts are disorders of the whole*
- ▶ A broader definition of “Disorder\_of\_lung”
  - ▶ Disorder THAT has\_locus SOME (Lung OR is\_clinical\_part\_of SOME Lung)

Equivalent Class (Necessary & Sufficient Criteria) —

**Disorder**  
has\_locus some (Lung or (is\_clinical\_part\_of some Lung))

Subclass Of (Necessary Criteria) —

Inferred/Inherited anonymous descriptions (Necessary criteria) —

▼ Disorder\_of\_lung

- ▼ Inflammation\_of\_lobe\_of\_lung
  - Lobar\_pneumonia
- ▼ Pneumonitis
  - ▶ Pneumonia

- ▶ Almost OK, but still Inflammation of lobe of lung is not a pneumonitis

# Make the pattern consistent

- ▶ Redefine Pneumonitis  
“An inflammation of the lung or any clinical part of the lung of the lung”

▶

### Class Annotations: Pneumonitis

Property	Value
comment	Pneumonitis is an inflammation of the Lungs or any part of the lungs

### Class Description: Pneumonitis

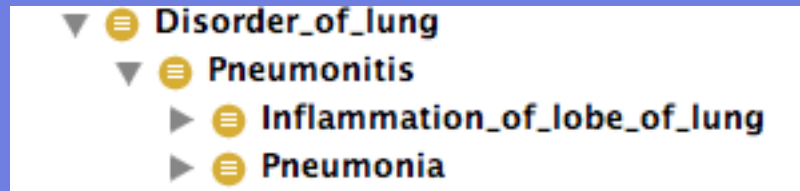
Equivalent Class (Necessary & Sufficient Criteria) —

**Inflammation**  
**has\_locus some (Lung or (is\_clinical\_part\_of some Lung))**

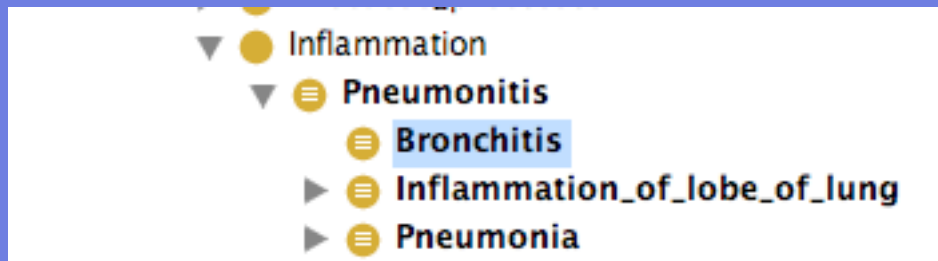
Subclass Of (Necessary Criteria) —

Inferred/Inherited anonymous descriptions (Necessary criteria) —

# Almost correct, but...

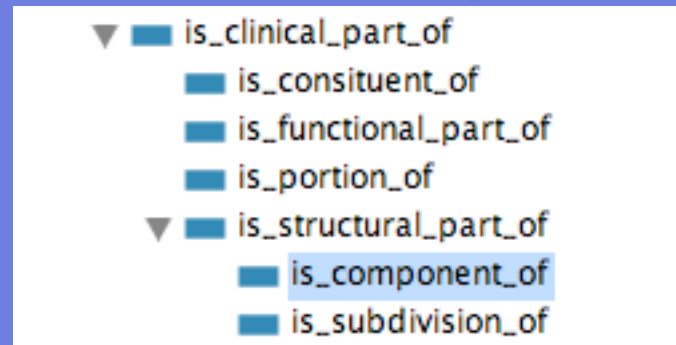


- ▶ What about “Bronchitis” ?
  - ▶ An inflammation of the bronchi (or any of their parts)
    - ▶ Try it and see.



- ▶ Definition of “Pneumonitis” is now too broad
- ▶ Not just any part of the lung, but the “subdivisions” of the lung
- ▶ lobes, quadrants, bases, apices, etc.

# The property hierarchy allows multiple views



- ▶ The bronchus is a “component” of the lung
- ▶ The lobe is a subdivision of the lung
- ▶ Redefine pneumonitis as an inflammation of the lung or a *subdivision* of the lung

▶ **Class Description: Pneumonitis**

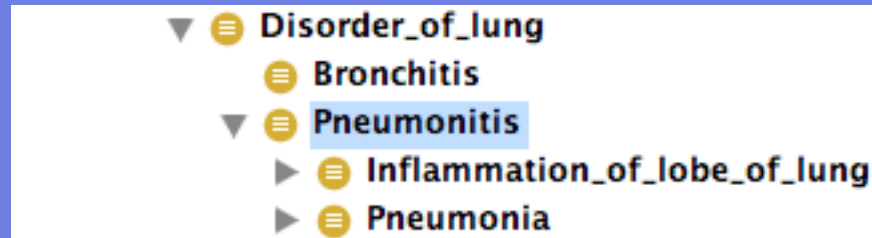
Equivalent Class (Necessary & Sufficient Criteria) \_\_\_\_\_

**Inflammation**

**has\_locus** some (Lung or (is\_subdivision\_of some Lung))

Subclass Of (Necessary Criteria) \_\_\_\_\_

# Now reclassify



- ▶ Bronchitis is now a disorder of the lung ( “lung disease”) but not a pneumonitis
  - ▶ As required.

# Clinical partonomy and pleuritis

- ▶ To an anatomist,
  - ▶ the pleura are different organs from the lungs
- ▶ To a clinician,
  - ▶ “Pleuritis” should be classified as a “Lung disease” or “Disorder of the lung”
    - ▶ “Pleuritis” - Inflammation of the pleura
- ▶ The Pleura
  - ▶ *function* as part of the lung
  - ▶ even though they are not *physically* part of the lung
    - ▶ The property hierarchy copes with both views.

- is\_clinical\_part\_of
  - is\_consistent\_of
  - is\_functional\_part\_of
  - is\_portion\_of
- is\_structural\_part\_of
  - is\_component\_of
  - is\_subdivision\_of

- ▶ Anything that is structurally a part of something is a clinical part of it
- ▶ Anything that is functionally a part of something is a clinical part of it
- ▶ etc.
- ▶ BUT NOT VICE VERSA.



# Also affects modularity

- ▶ We have chosen to model functional parts with physiology rather than with anatomy
  - ▶ To stick with the FMA view as far as possible in the Anatomy module.
- ▶ So we add the fact that the pleura are functionally part of the Lung in the `physiologic_processes` module rather than the anatomy module
  - ▶ Might even have a separate functional module
    - ▶ We can add information to a class in a new module

The screenshot shows a class editor for 'Pleura'. It is divided into two main sections: 'Class Annotations: Pleura' and 'Class Description: Pleura'.

**Class Annotations: Pleura**

Property	Value
comment	The covering of the lung.  (In FMA it is a separate organ rather than organ part for embryological reasons.)
comment	Treated as functionally part of the lung for pathophysiological purposes even though not structurally part of the lung for anatomical purposes.  See chapter on clinical anatomy in Burger et al.

**Class Description: Pleura**

Equivalent Class (Necessary & Sufficient Criteria) \_\_\_\_\_

Subclass Of (Necessary Criteria) \_\_\_\_\_

- Organ
- has\_pairedness **some** Paired
- is\_functional\_part\_of some Lung**

Inferred/Inherited anonymous descriptions (Necessary criteria) \_\_\_\_\_

Additions in *physiologic\_processes.owl*

# Create pleuritis and classify

**Class Annotations: Pleuritis**

Property	Value	Lang
comm...	Pleuritis - Inflammation of the Pleura or any of their subdivisions	

**Class Description: Pleuritis**

Equivalent Class (Necessary & Sufficient Criteria) —

**Inflammation**

**has\_locus some (Pleura or (is\_subdivision\_of some Pleura))**

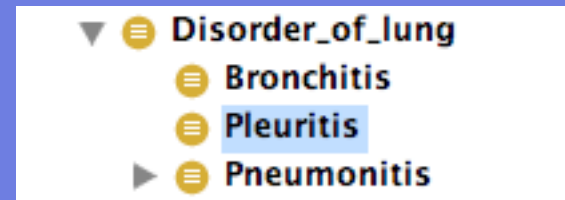
Subclass Of (Necessary Criteria) —

## ▶ Classify and check results

▶ A disorder of the lung but not a bronchitis or pneumonitis.

▶ as required

▶ *Anatomists & Clinicians can each have their own view*



# Normality and Negation

- ▶ What does it mean to be normal or abnormal?
  - ▶ To have a disease
- ▶ We implement two notions -
  - ▶ NonNormal - anything noteworthy
    - ▶ Pathological - requiring medical intervention
      - ▶ (*including “watchful waiting” or an active decision not to intervene*)
        - ▶ *GALEN used “Intrinsically pathological” but not needed in OWL*
- ▶ Basic rules
  - ▶ Pathological → nonNormal
  - ▶ Normal = NOT nonNormal
  - ▶ nonPathological = NOT pathological

# Normality and negation

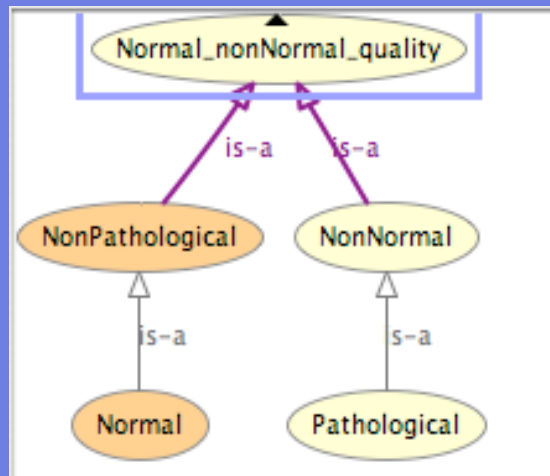
## ▶ Basic rules

- ▶ Pathological → nonNormal
- ▶ Normal = NOT nonNormal
- ▶ nonPathological = NOT pathological

## ▶ Remember

- ▶ subclassOf means “necessarily implies”
  - ▶ so Pathological is a subclass of nonNormal
- ▶ See the definitions of Normal-nonNormal\_quality in disorders.owl

## ▶ Let the classifier do the work...



# Defining “disease” or “disorder”

- ▶ Hard, probably futile
  - ▶ The words are used in many different ways
  - ▶ Things referred to cross ontological boundaries
    - ▶ Lesions - e.g tumours
    - ▶ Processes - e.g. infection or inflammation
    - ▶ Qualities - e.g. obstruction, malformation, elevation, ...
- ▶ Best just to say what is pathological
  - ▶ *let the classifier gather them up*
- ▶ Also classify along multiple dimensions
  - ▶ *include as many abstractions as are useful, no more and no less*

# Example from tiny tutorial ontology

- ▼  Disorder
  - ▼  Disorder\_of\_anatomical\_structure
    - ▶  Lobar\_pneumonia
    - Ulcer\_of\_stomach
    - Ulceration\_of\_stomach
  - ▼  Disorder\_of\_lung
    - ▶  Pneumonitis
    - Fever
  - ▼  Pathological\_process
    - ▶  Erosion\_process
    - Hypertension
    - Infection
    - ▶  Infectious\_processes
    - ▶  Inflammation
    - ▶  Metabolic\_disorder
  - ▼  Pathological\_structure
    - ▼  Erosion\_lesion
      - ▶  Ulcer
    - ▼  Injury
      - Abrasion
      - Fracture
      - Laceration
    - Malignant\_tumour

# Note

- ▶ Commented version of my\_ontologies is in
  - ▶ [Specific\\_disorders.owl](#)
    - ▶ Make that the active ontology and compare

# Situations & Codes

- ▶ The package unit of information in electronic health records is a “Situation”
  - ▶ A patient as observed by a clinician at a time in a setting
- ▶ Adding a code to an event in a record is to assert that the event is an instance of that kind of situation
  - ▶ The event has at least all of the assertions comprising the individual codes
  - ▶ Allows easy expression of negation and classification of results with recognition of equivalence



# Example: Head Trauma with/without intracranial bleeding with/ without Skull Fracture

## Class Description: Head\_trauma\_situation

Equivalent classes 

- Situation  
that includes some Head\_trauma

## Class Description: Head\_trauma\_with\_sk

Equivalent classes 

- Situation  
that includes some Head\_trauma  
and includes some Skull\_fracture

## Class Description: Head\_trauma\_without\_skull\_fracture\_with\_subdural\_bleed\_

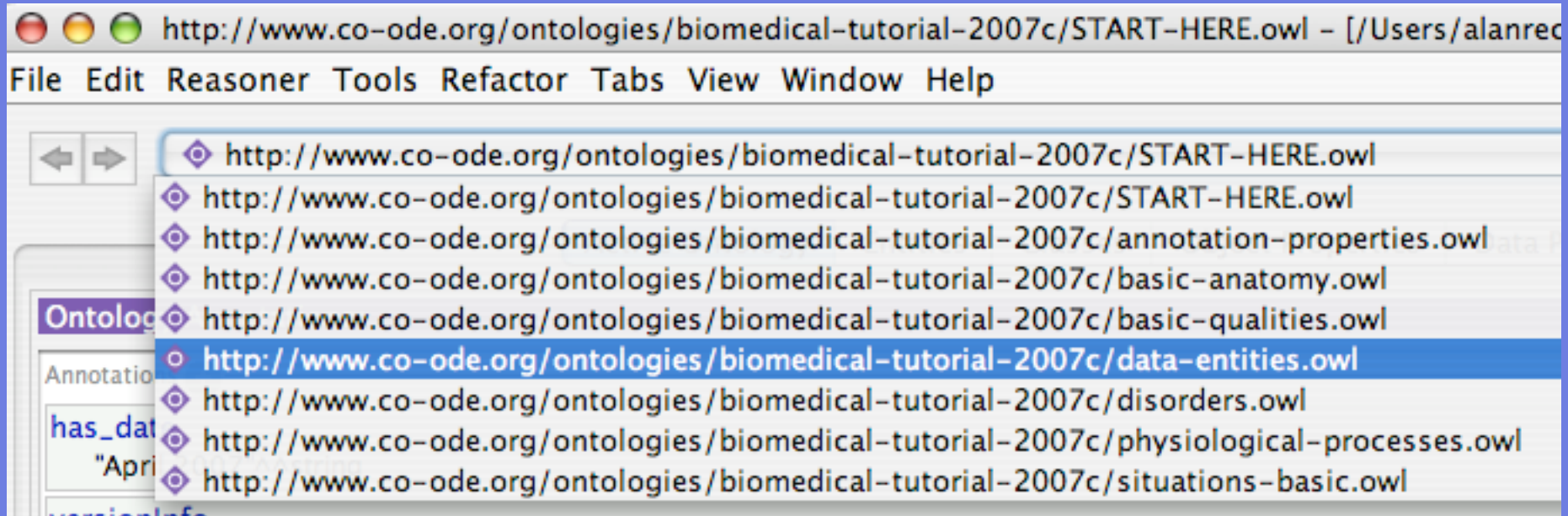
Equivalent classes 

- Situation  
that includes some Subdural\_bleed  
and not (includes some Skull\_fracture)  
and includes some Head\_trauma



# Quantities and Units

- ▶ A pervasive issue, so we shall take a brief look now
- ▶ Make *data-entities.owl* the active ontology
  - ▶ Right-click or cmd-click in the OWLViz View
  - ▶ Select from the menu at the top of the screen

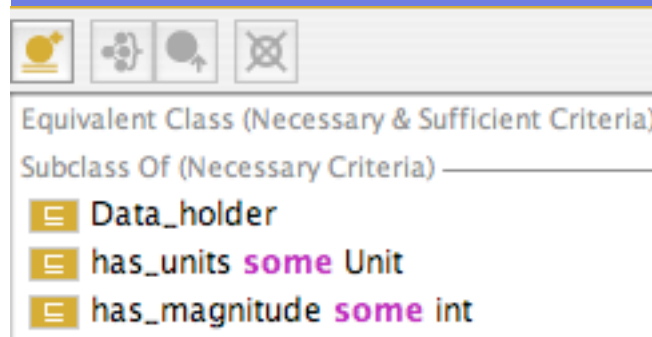
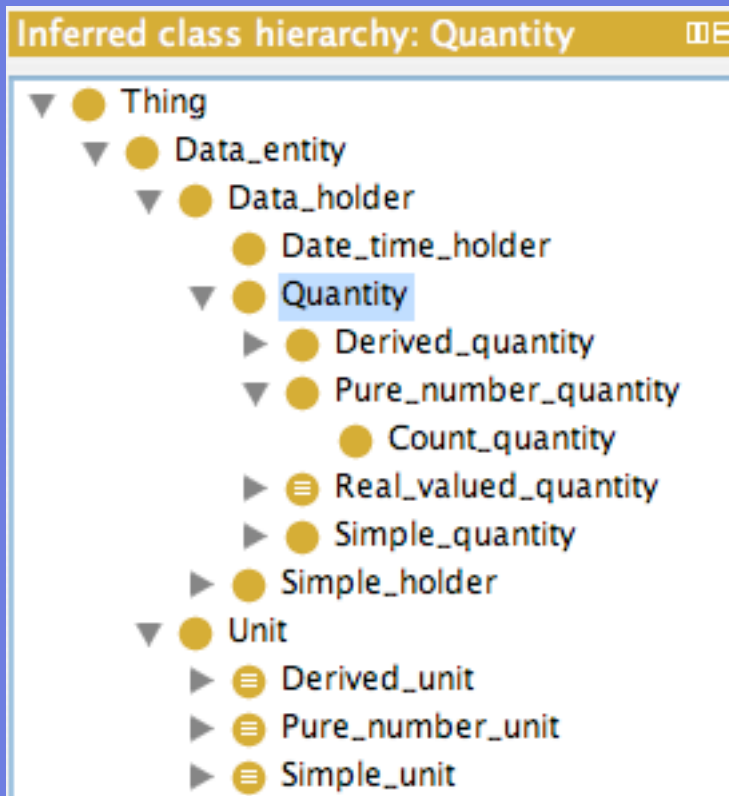


# Quantities

- ▶ As real as numbers, matrices, or any other mathematical structure
  - ▶ “Naked” numbers rarely suitable for healthcare IT
    - ▶ Too much chance of error
      - ▶ *Mars landers have failed and patients have died*
  - ▶ Distinguish “naked numbers” from “pure numbers” - e.g. percentages, universal constants etc
- ▶ Quantities have
  - ▶ magnitude
  - ▶ units
  - ▶ dimension
  - ▶ dimension and units must be compatible
  - ▶ dimension is usually indicated by units
- ▶ Time and Duration
  - ▶ Note that Dates and times and temporal intervals (temporal deictics) are not quantities
    - ▶ The difference between two lengths is a length
    - ▶ The difference between two times is a duration
    - ▶ Date-times and temporal intervals require separate mechanisms
  - ▶ Duration is a quantity.

# Simple structure in tutorial ontology

- ▶ Dimension implicit in classification of quantities
- ▶ unit an object
  - ▶ Classifier forces subsumption of units to track subsumption of quantities
- ▶ magnitude a number
  - ▶ “int” is artifact of current state of classifier



*should be number*

# Typical quantities

- ▶ Specific value
  - ▶ Concentration\_quantity THAT  
has\_units SOME mg\_per\_L  
has\_magnitude VALUE 140
- ▶ Value range (OWL 1.1 / new version only)
  - ▶ Concentration\_quantity THAT  
has\_units SOME mg\_per\_L  
has\_magnitude SOME int[ >=130, <=150]
- ▶ NB units mg\_per\_L will cause quantity to be classified as a *Mass\_concentration\_quantity*
  - ▶ Try it in DL query tabl.

# DL Query for previous

Query:

Query (class expression)

Concentration\_quantity **that**  
has\_units **some** mg\_per\_L and  
has\_magnitude **value** 140

Execute

Query results

Super classes —

- Mass\_per\_volume\_concentration\_quantity
- Concentration\_quantity

Equivalent classes —

Subclasses —

Individuals —

# Quantities and Units

## *which is “primitive”?*

### ▶ Requirements

- ▶ Avoid maintaining the hierarchies of quantities and units separately
- ▶ Be able to determine the legal units for a quantity
- ▶ Be able to “co-erce” the quantity according to the units

### ▶ Our solution

- ▶ All classes of units are asserted in a flat list
- ▶ Defined following the pattern

- ▶ Unit\_class = *is\_units\_of* SOME Quantity\_class  
→ *is\_units\_of* ONLY Quantity\_class

- ▶ *“Any unit for this class of quantity must be of this kind and only of this kind”*

### ▶ Query for unit for a quantity

- ▶ Unit THAT *is\_units\_of* SOME quantity\_class

### ▶ Quantity that uses units

- ▶ Quantity that *has\_units* SOME Unit\_class



# DL Queries

## Query:

Query (class expression)

Unit **that** is\_units\_of **only** Mass\_per\_volume\_concentration\_quantity

Execute

## Query results

Super classes

☰ **Derived\_unit**

Equivalent classes

☰ **Mass\_per\_volume\_concentration\_unit**

Subclasses

● **mg\_percent**

● **mg\_per\_L**

Individuals

## Query:

Query (class expression)

Quantity **that** has\_units **some** mg\_per\_L

Execute

## Query results

Super classes

● **Mass\_per\_volume\_concentration\_quantity**

● **Concentration\_quantity**

Equivalent classes

Subclasses

Descendant classes

Individuals

# Example of use

- ▶ “Hemoglobin 13 mg%”
  - ▶ Serum\_hemoglobin THAT
    - has\_value SOME (Concentration THAT
      - has\_magnitude VALUE 13
      - has\_units SOME mg\_per\_cent

# Extensions to quantities

- ▶ Compatible java packages for units and conversion.
  - ▶ Really ought to disappear into datatypes
    - ▶ but it seems unlikely, so...

# Summary: Building Ontologies in OWL-DL

- Start with a taxonomy of primitive classes
  - Should form pure trees
  - Remember, to make disjointness explicit
- Use definitions and the classifier to create multiple hierarchies
  - Use existential (`someValuesFrom`) restrictions by default
  - Things will only be classified under defined classes
- Be careful with
  - Open world reasoning
    - Use closure axioms when needed
  - “some” and “only” – `someValuesFrom/allValuesFrom`
  - domain and range constraints
  - making disjoint explicit

# Summary

- ▶ Knowledge is fractal
  - ▶ Enumeration is never ending
    - ▶ The power of logic / OWL is composition and classification
- ▶ Normalise ontologies for re-use and maintenance
  - ▶ Build DAGs (nets) out of Trees using classification
  - ▶ Use Modules to separate views and then bind them
- ▶ Diseases of the parts are diseases of the whole
  - ▶ ... but must be careful
- ▶ The property hierarchy can be used to support multiple views
- ▶ Some notions defy definition - e.g. “Disease”
  - ▶ When in doubt describe, classify and collect result bottom up
- ▶ Much more in the comments in the tutorial ontology
- ▶ and remember: ***“Ontologies are just one part of a system”***