MANCHESTER 1824

The University of Manchester

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
www.clinical-escience.org
www.opengalen.org

NIBHI

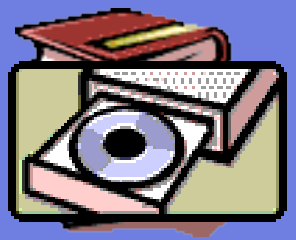
Tools and downloads

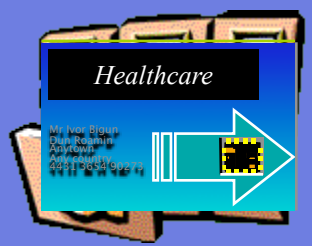
- Protege4Alpha
  - protege.stanford.edu
  - http://protege.stanford.edu/download/prerelease-alpha/protege-bin-4.0-alpha.zip
- GraphViz - required for OWLViz
  - http://www.graphviz.org/
- Tutorial handouts and Ontologies
  - http://www.cs.man.ac.uk/~rector/tutorials/Biomed-Tutorial-2007b-dagstuhl/

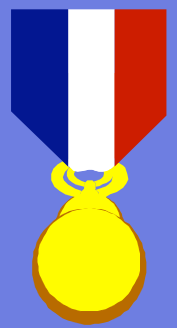
2

Where I come from

Clinical Terminology ← Data Entry ← Clinical Record ← Clinical research & Decision Support ← Best Practice







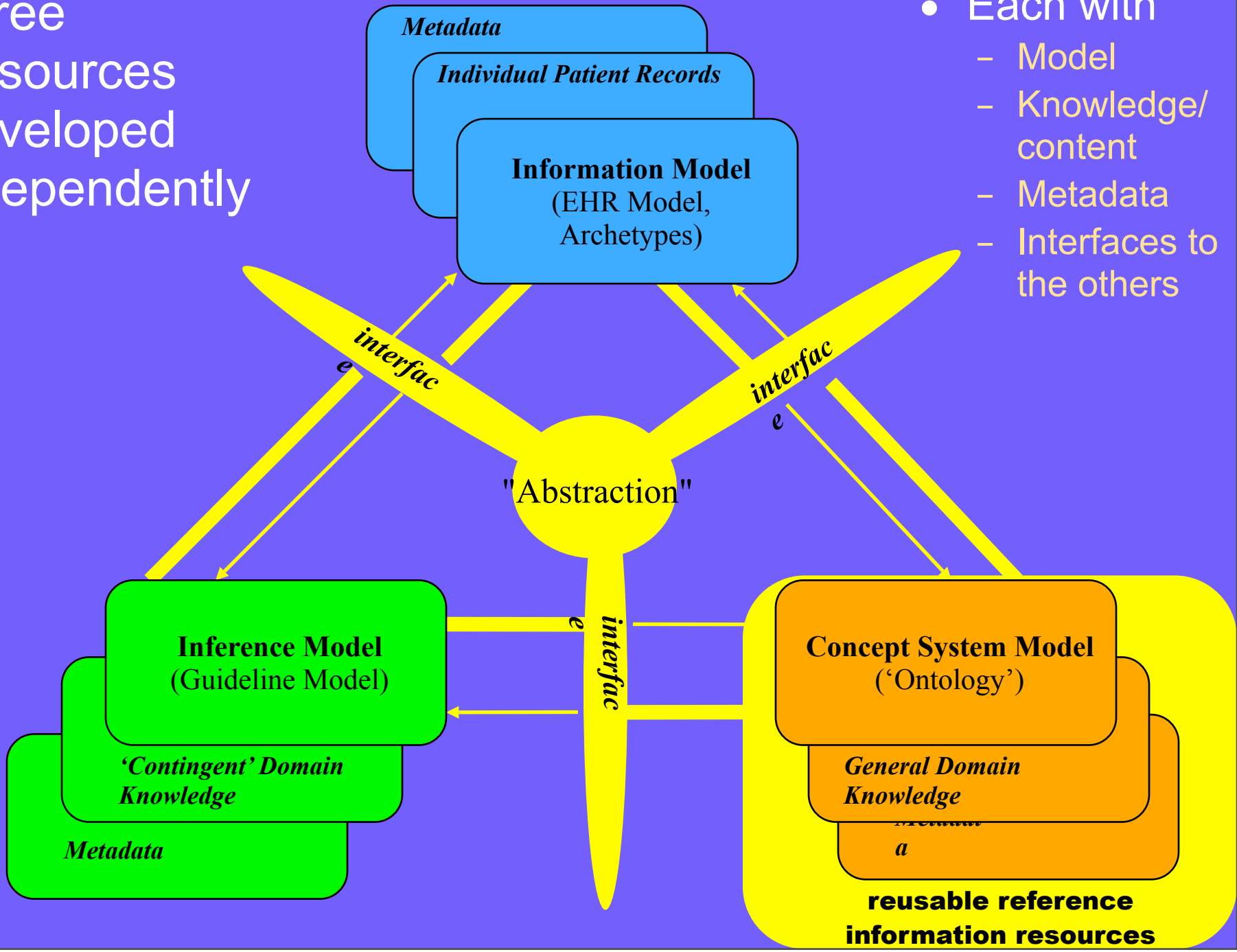
GALEN Ontologies & Description logics → Data Entry → Electronic Health Records → Clinical research → Decision Support & Knowledge Presentation → **Best Practice**

3

“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 Scalabilty / 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***

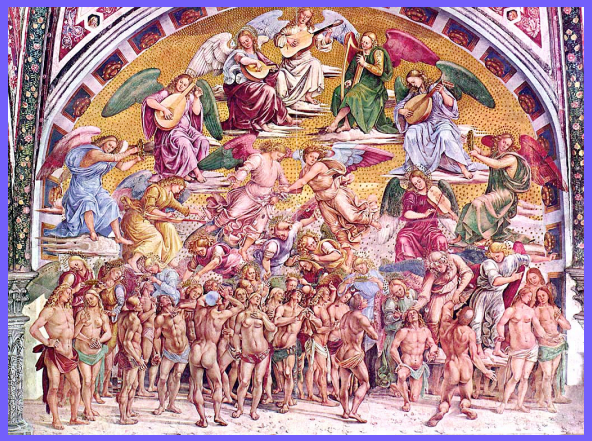
Three Resources Developed independently



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

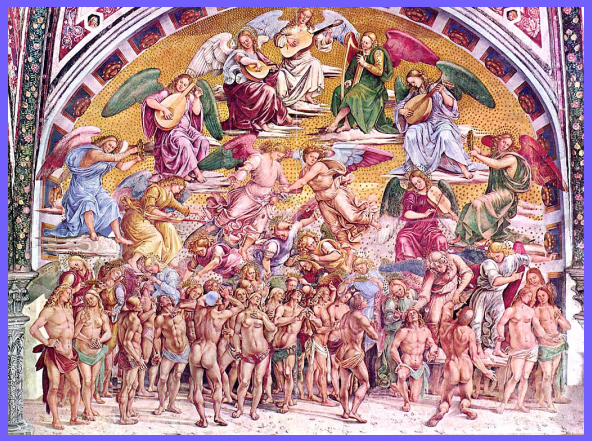
By way of User Centred Design

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
  - …

7



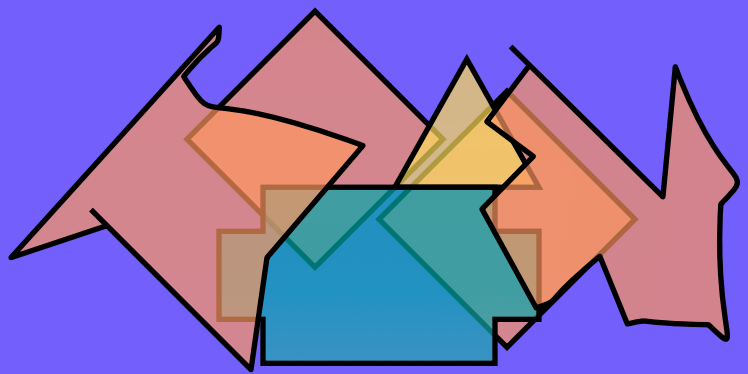
- Problem space
  - Answer questions
  - Advising on actions
  - Hazard monitoring
  - Creating forms
  - Discovering resources
  - Constraint actions
  - Assess risk
  - …

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

8

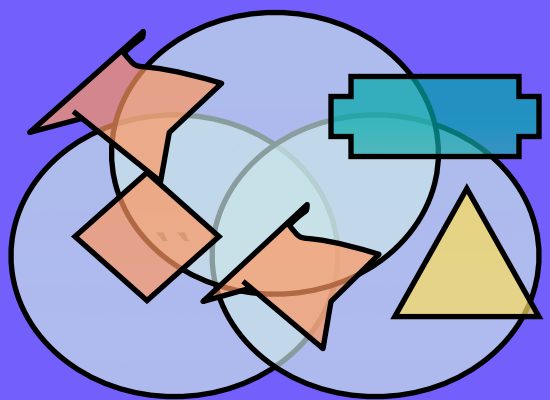
Problem space & solution space

Problem space



Guidelines, Patterns, Tools

Solution space



9

Some Recent User Experiences

- Two companies building clinical systems
  - Intelligent forms creation and constraint management
    - Mastering a combinatorial explosion of ≥10^7
  - Workshops, Training and subcontracting
- NHS National Programme for IT:
  - Additional constraints to UML and between UML and terminology/ontology
    - Different implementation of the standard don't fit
  - Managing SNOMED
    - Large ≥450K ters DL based terminology
- Biologists research groups
  - Pathways
  - Epidemiology / Clinical Trials
  - Normalisation of Gene Ontology
  - Discovery of Phosphatases and other Protein activity
  - Animal behaviour and biological images
-

10

Topics for today

- Normalisation & Why Classify?
  - Why use a computable subset of logic?
- Modularisation - doing it in layers
- Anatomy, parts and Disorders
  - Pneumonitis and pneumonias
  - A disorders of the lung
- Quantities and Units
- Normal, NonNormal & Pathological
  - Using negation

11

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*

12

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:

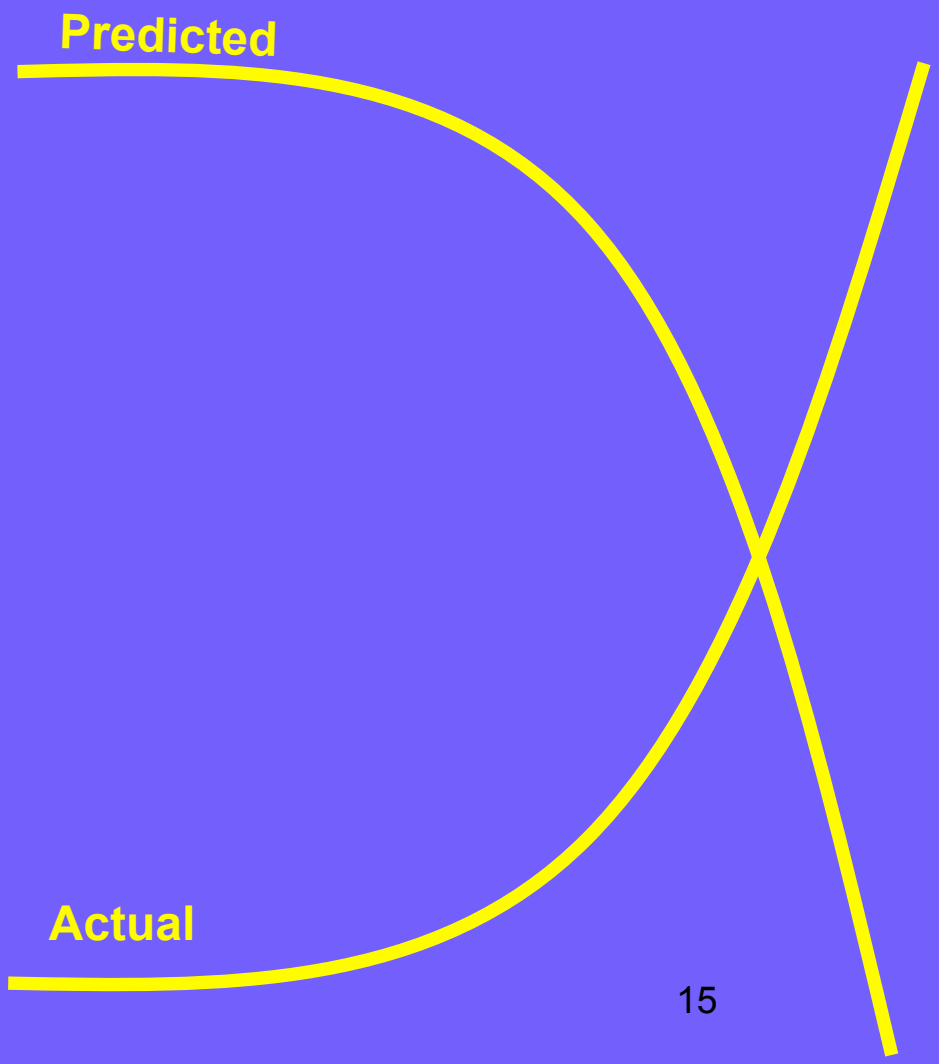
Enumeration doesn't scale

14

The scaling problem: The combinatorial explosion

- It keeps happening!
  - "Simple" brute force solutions do not scale up!

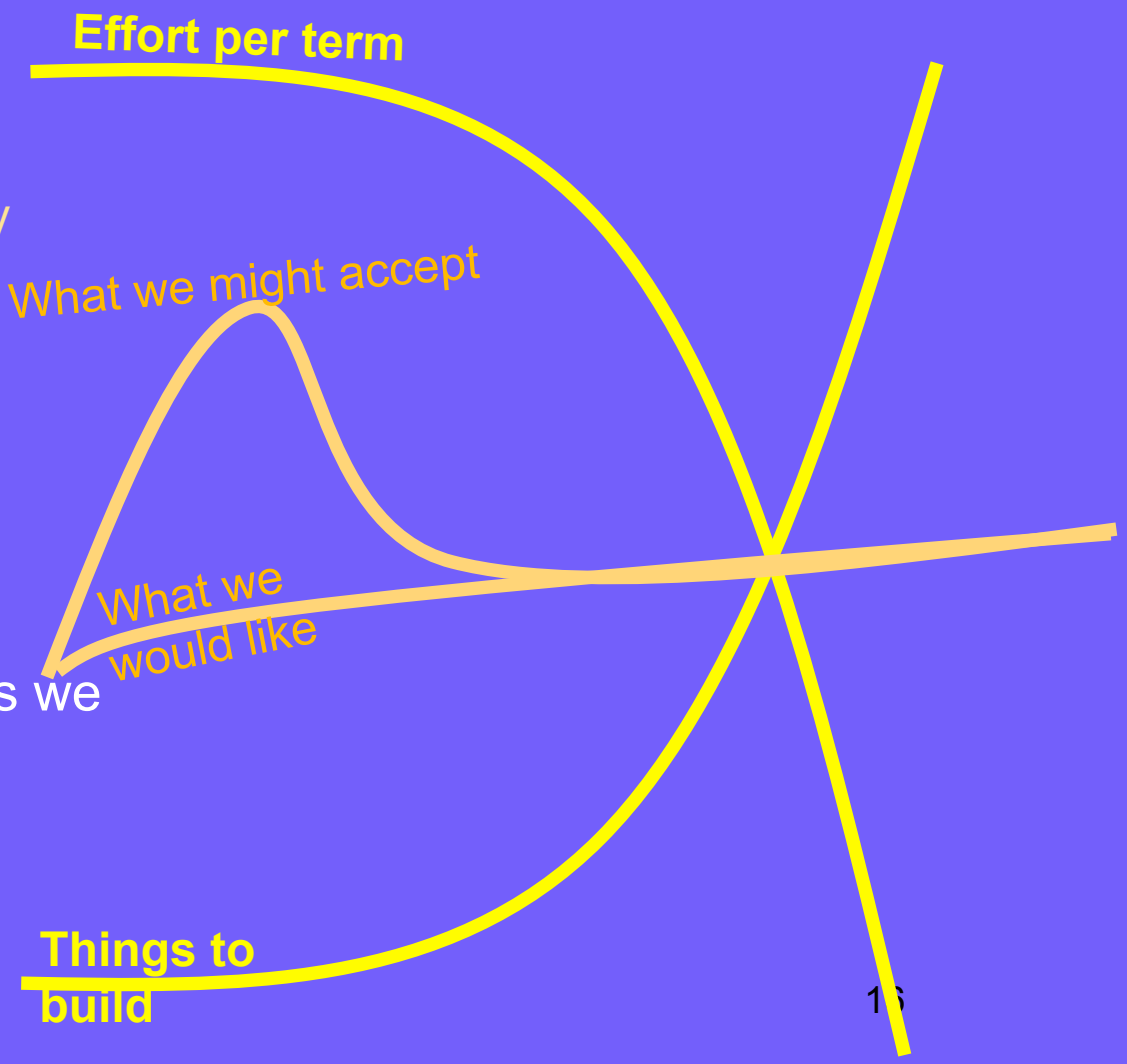
- Conditions x sites x modifiers x activity x context→
  - *Huge number of terms to author*
  - *Software CHAOS*



15

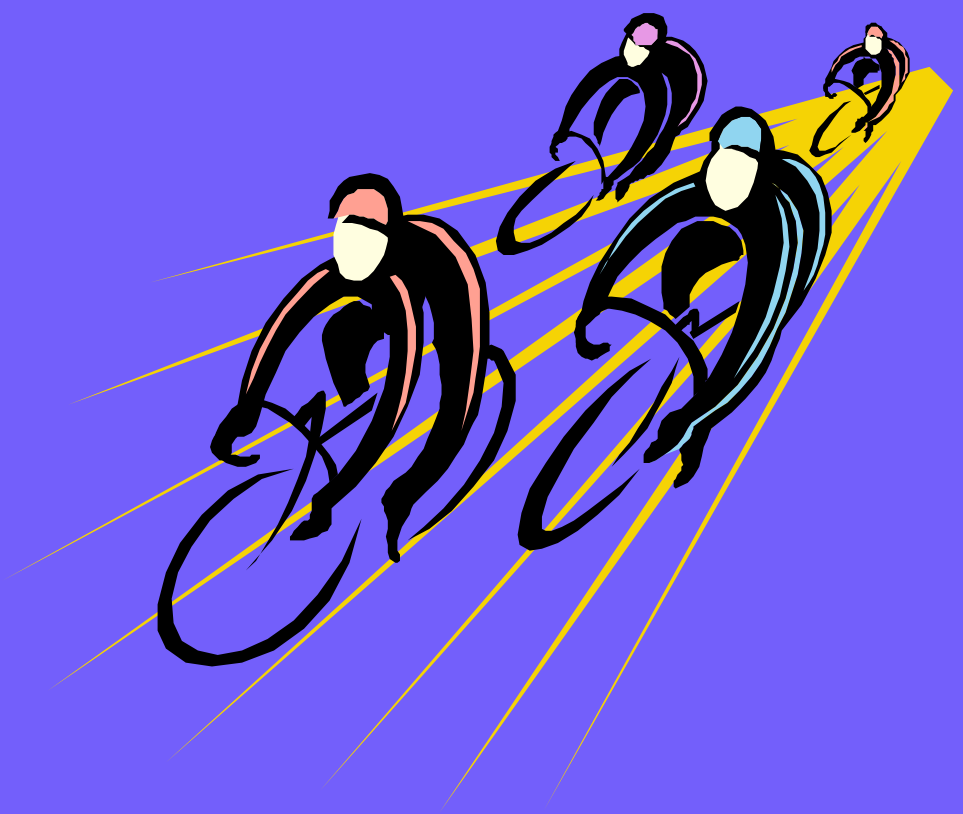
Combination of things to be done & time to do each thing

- 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 ……



17

1999 ICD10: 587 codes

- V31.22 Occupant of three–wheeled motor vehicle injured in collision with pedal cycle, person on outside of vehicle, nontraffic accident, while working for income
- W65.40 Drowning and submersion while in bath–tub, street and highway, while engaged in sports activity
- X35.44 Victim of volcanic eruption, street and highway, while resting, sleeping, eating or engaging in other vital activities

18

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

19

Conceptual Lego… it could be...

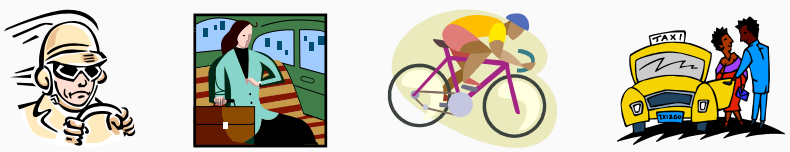
Structured Data Entry

File Edit Help

Cycling Accident

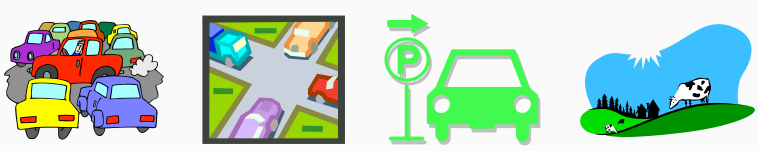
What you hit

Your Role



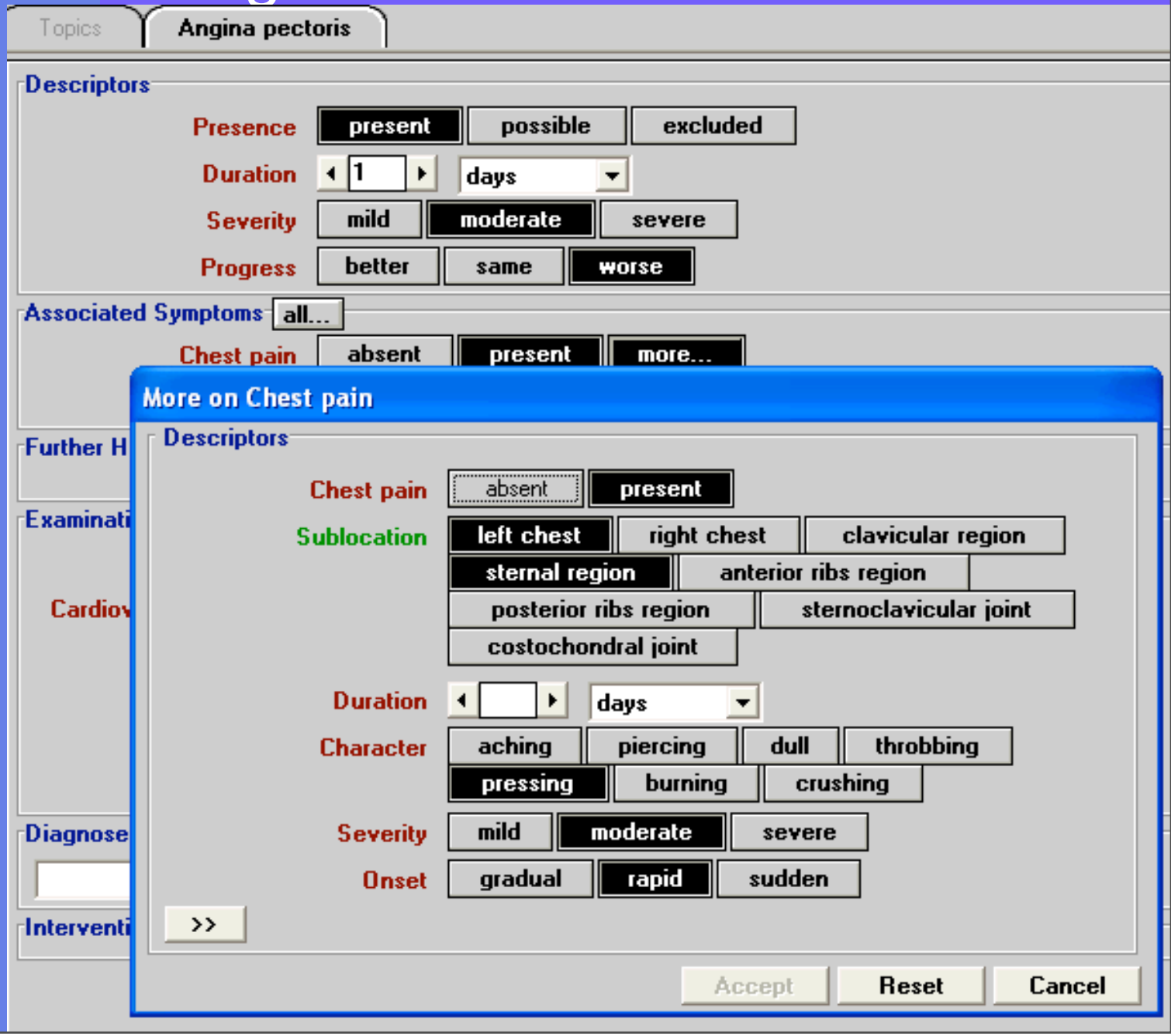
Activity

Location

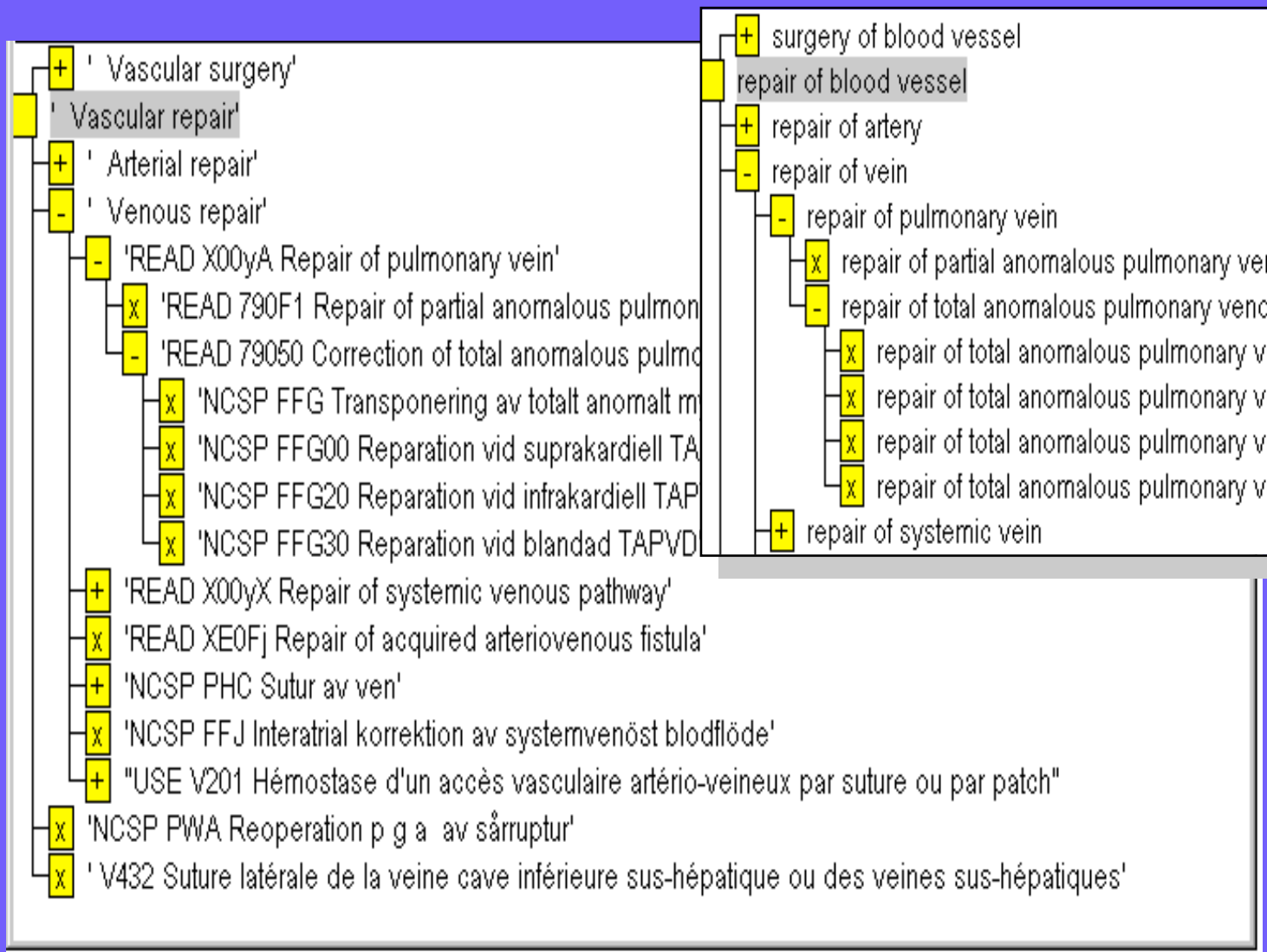


20

Intelligent Forms

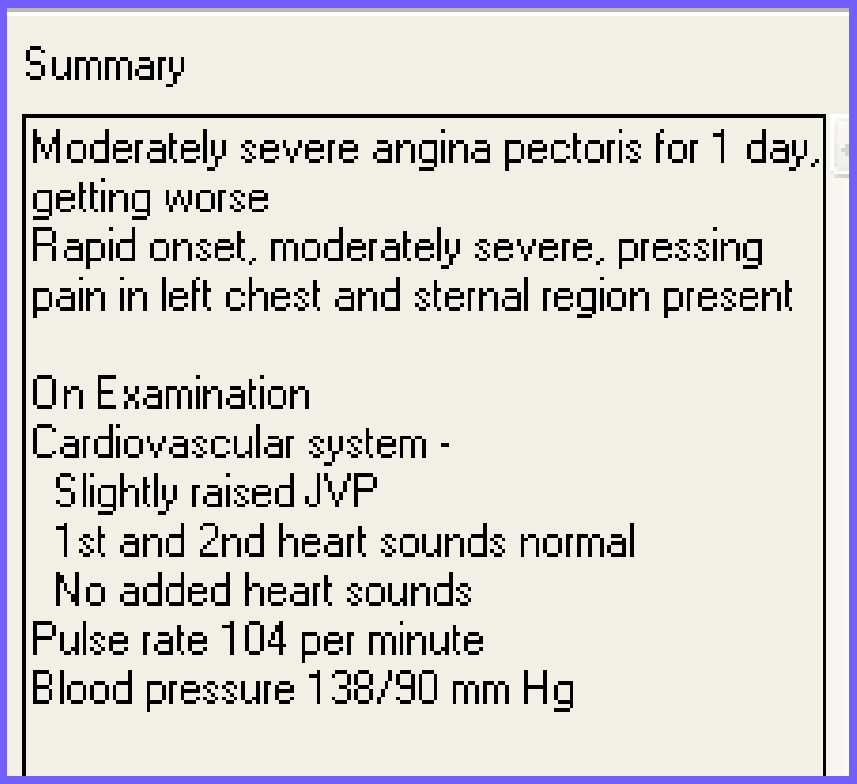


Integrating rather than Cross Mapping



22

And generate it in language



23

Supports Loosely coupled distributed ontology development

global cycle & work at centre

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

polysacharide

cell

expression

Lung

infection

inflammation

bacterium

virus

25

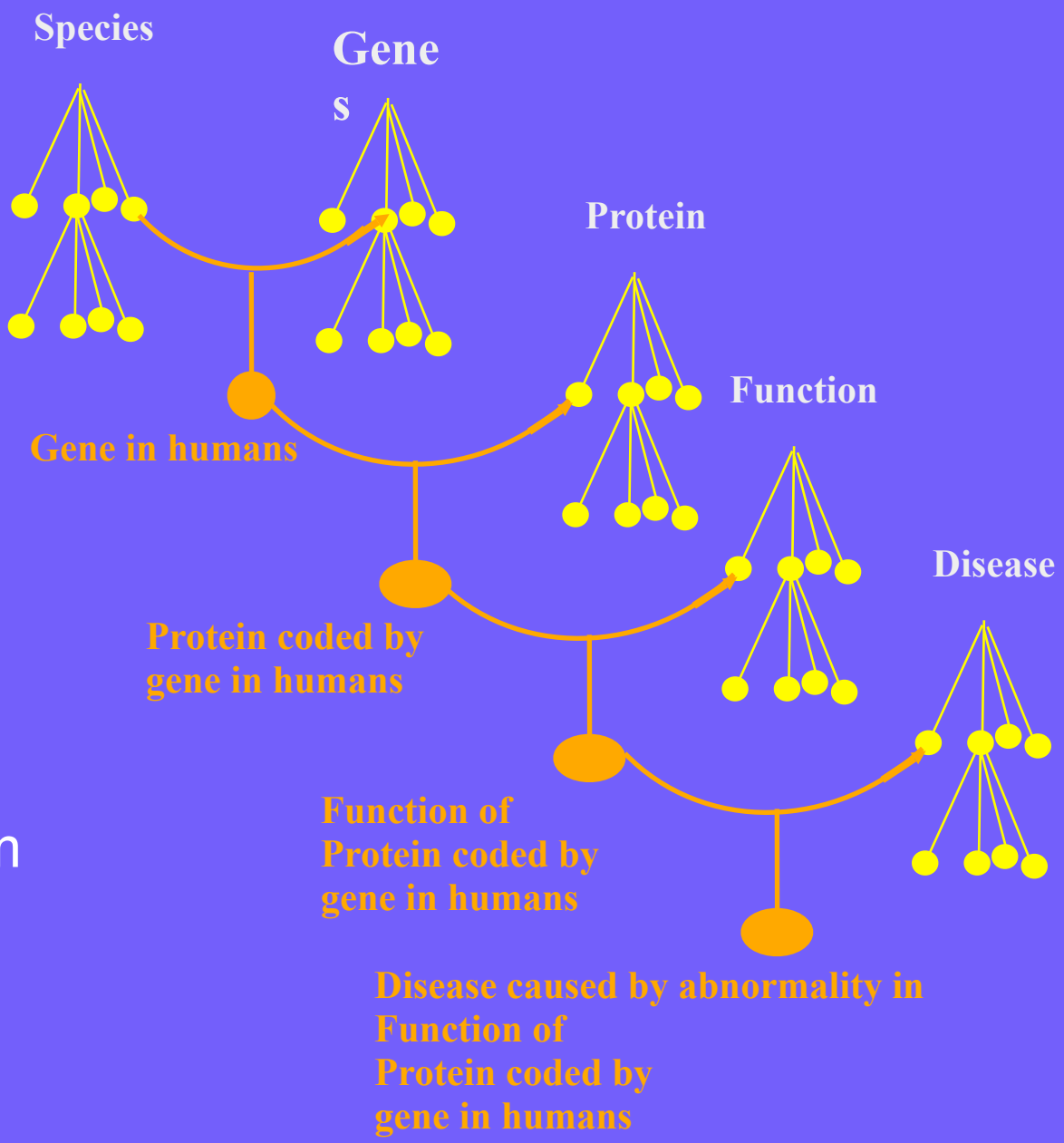
Logic as the clips for "Conceptual Lego"

"***SNPolymorphism*** of ***CFTRGene*** causing ***Defect in MembraneTransport*** of ***Chloride Ion*** causing ***Increase*** in ***Viscosity*** of ***Mucus*** in ***CysticFibrosis***…"

"***Hand*** which is ***anatomically normal***"

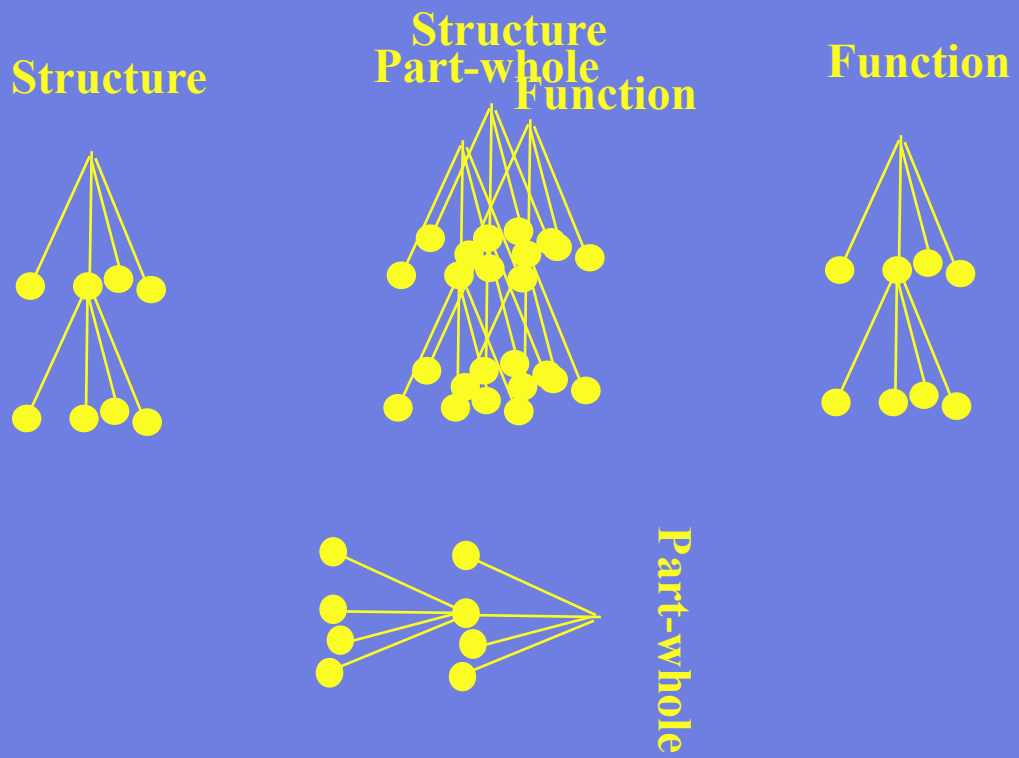
26

Build complex representations from modularised



27

Normalising (untangling) Ontologies



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

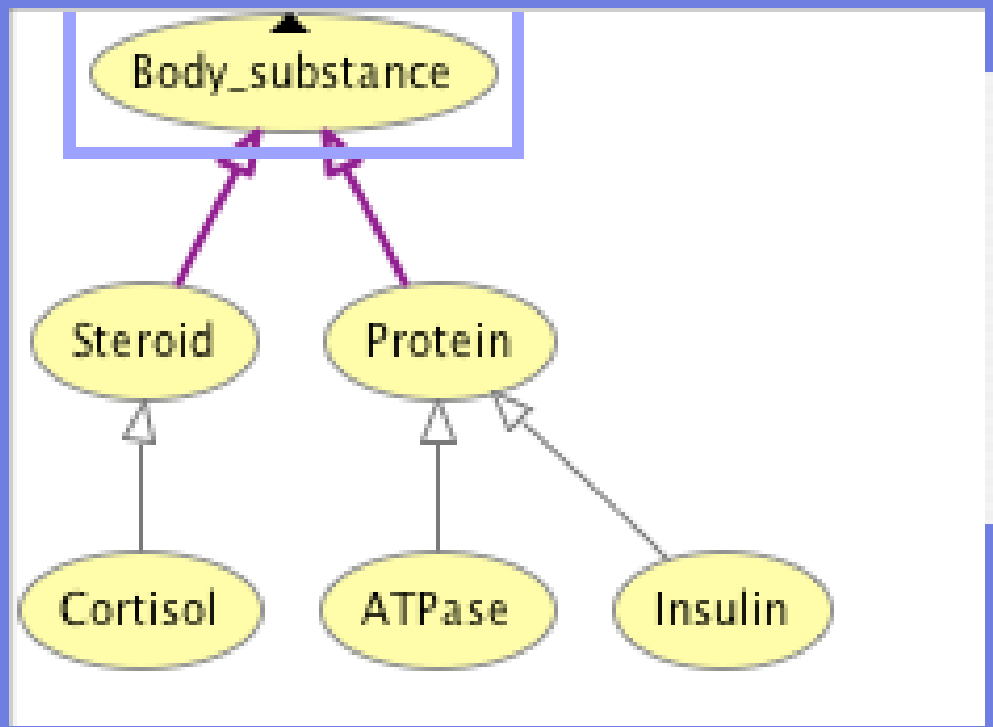
Using a classifier to make life easier

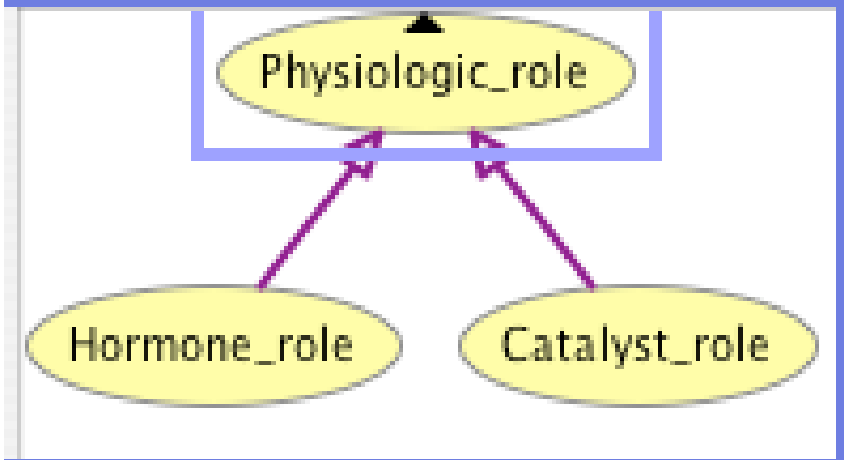
- Substance
- - Protein
- - - Insulin
- - - ATPase
- Steroid
- - Cortisol

- PhsioloicRole
- - HormoneRole
- - CatalystRole

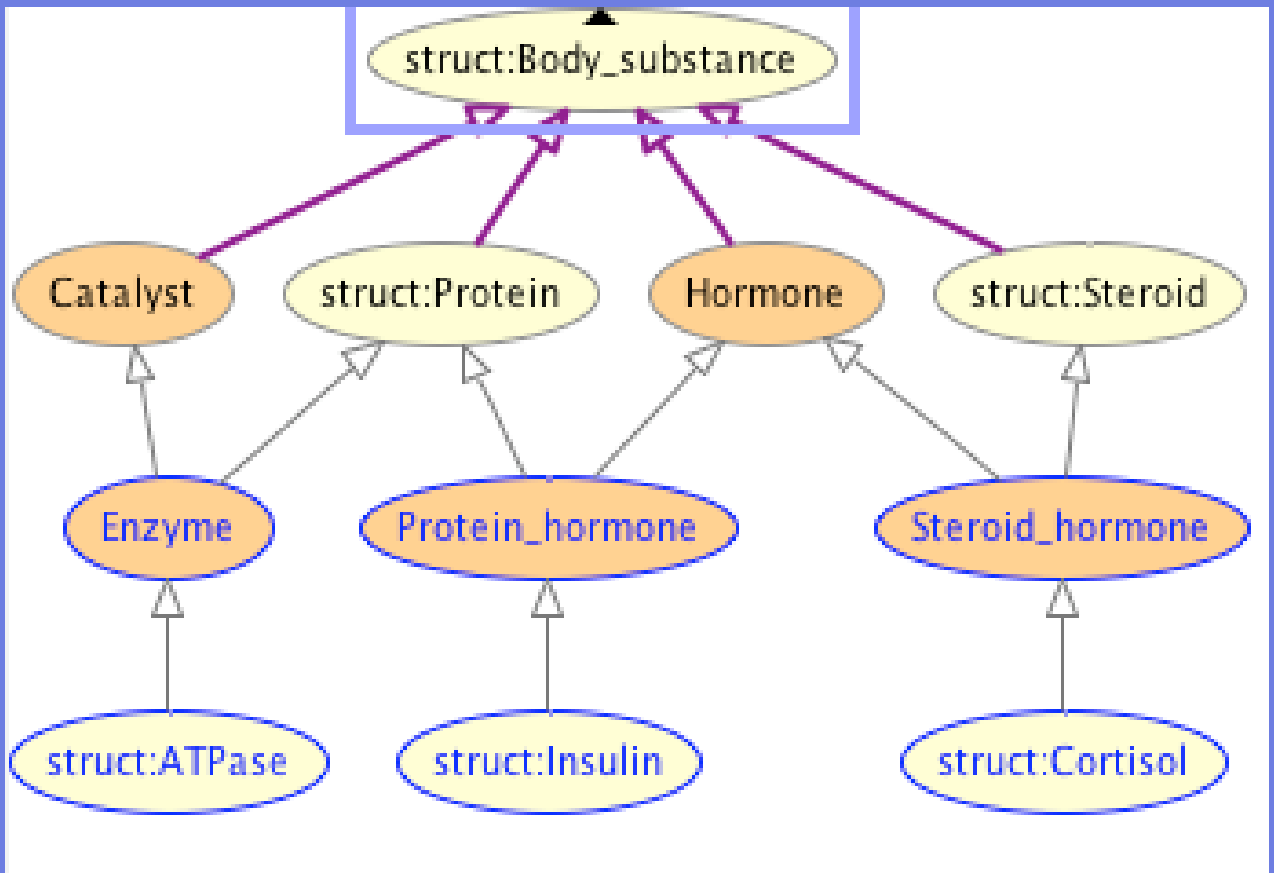
Substance
- Protein
- - ProteinHormone
- - - Insulin
- - Enzyme
- - - ATPase
- Steroid
- - SteroidHomone^
- - - Cortisol
-Hormone
- - ProteinHormone^
- - - Insulin^
- - SteroidHormone^
- - - Cortisol^
- Catalyst
- - Enzyme^
- - - ATPase^

Modularised into structure and function ontologies

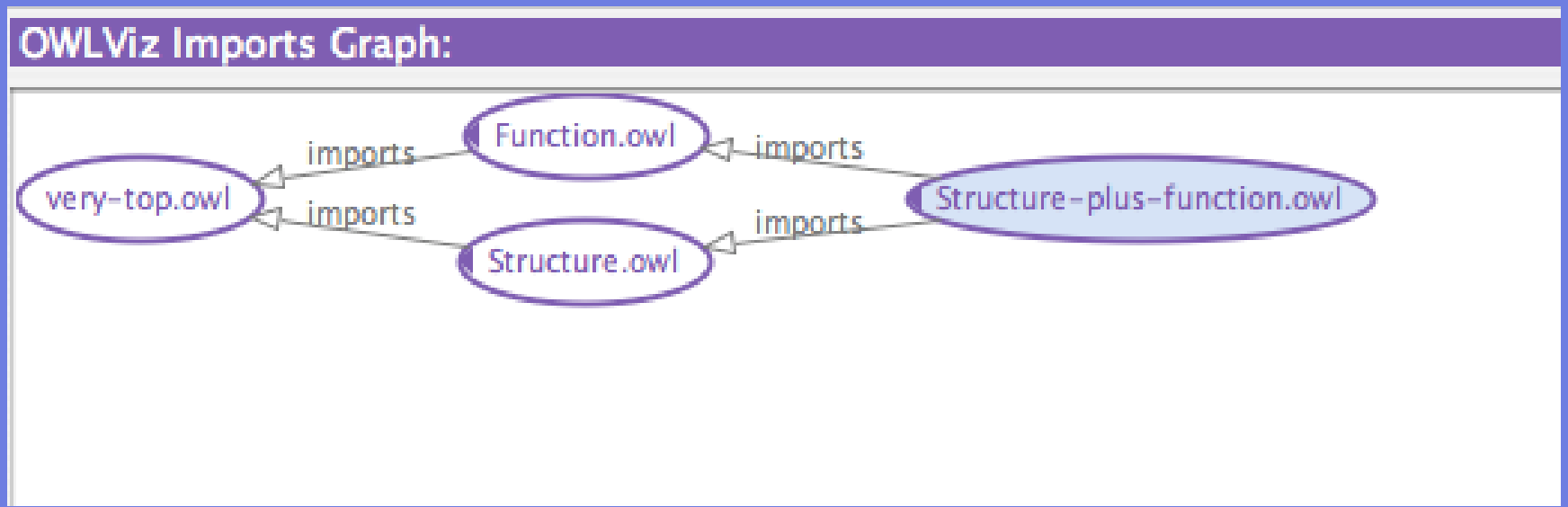




Unified ontology after classification



Module Structure



34

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: Parent & $\neg sub_1$ &…& $\neg sub_{n-1}$ → 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 <u>locally</u> disjoint & closed

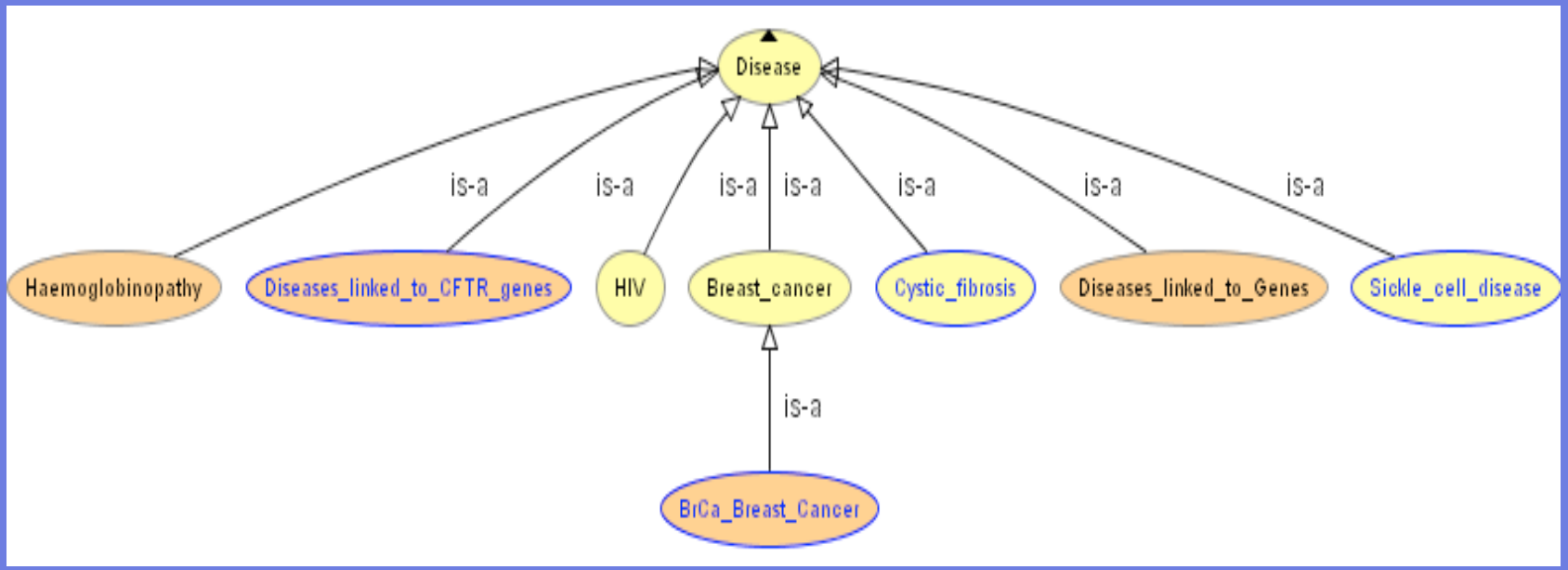
- individual values must be disjoint
  - but can be hierarchical
    - e.g. “very hot”, “moderately severe”
- Each list can be guaranteed to be complete
  - Can infer Parent & ¬sub_1 &…& ¬sub_{n-1} → sub_n
- Value types themselves need not be disjoint
  - “being hot” is not disjoint from “being severe”
    - 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*

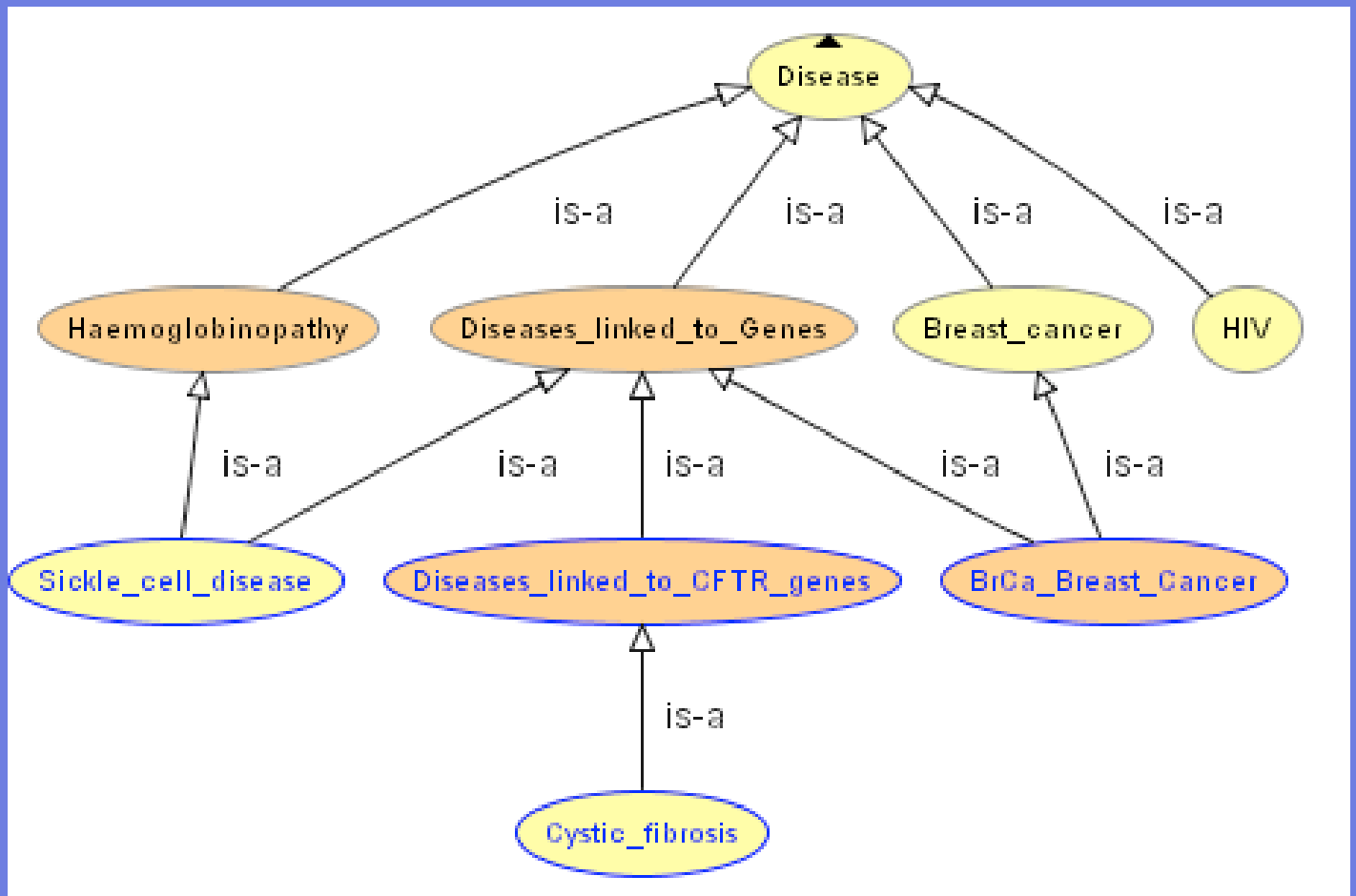
  No axiom should imply that a primitive is part of more than one branch of primitive skeleton
  - If all primitives are disjoint, any such axioms will make that primitive unsatisfiable
    - A partial test for normalisation:
      - *Create random conjunctions of primitives which do not subsume each other.*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

A real example: Build a simple treee

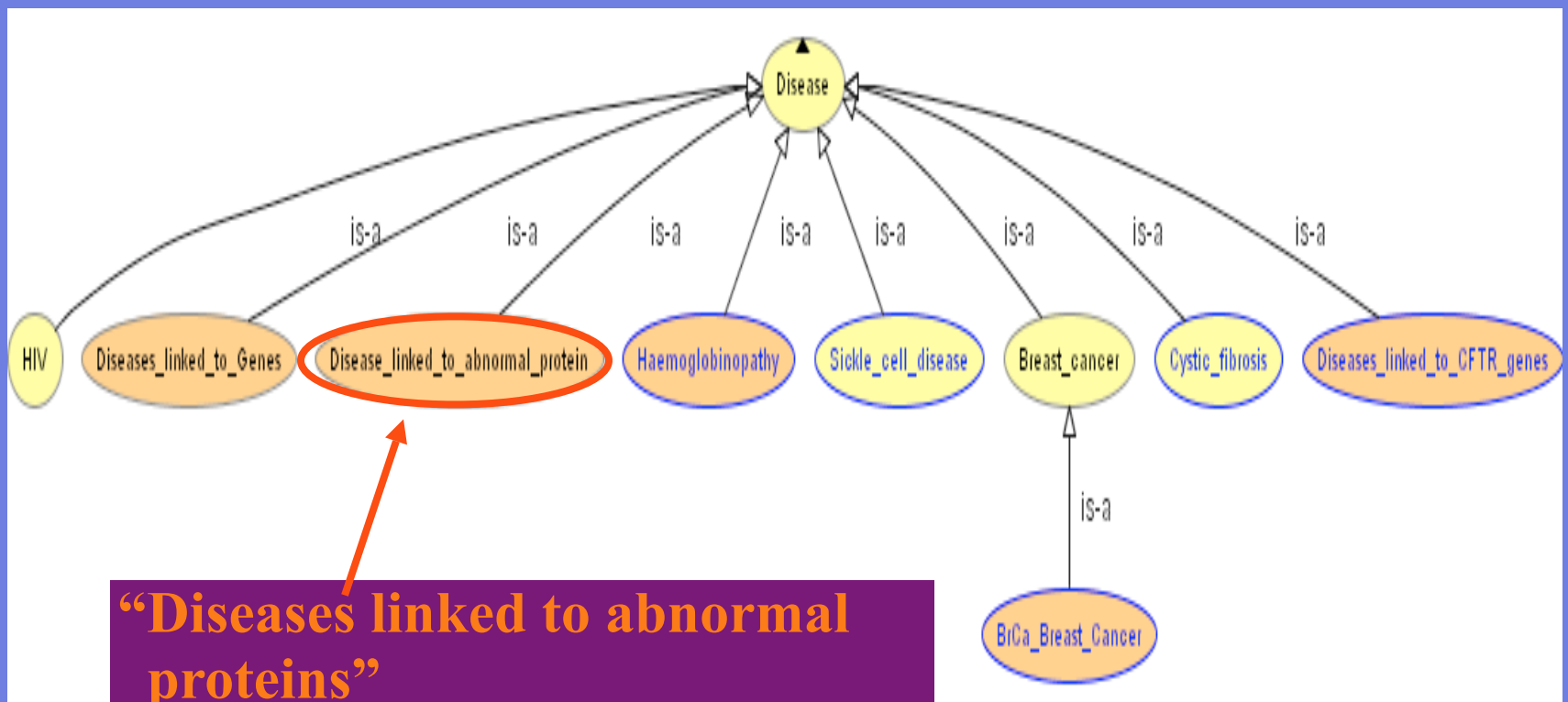


easy to maintain

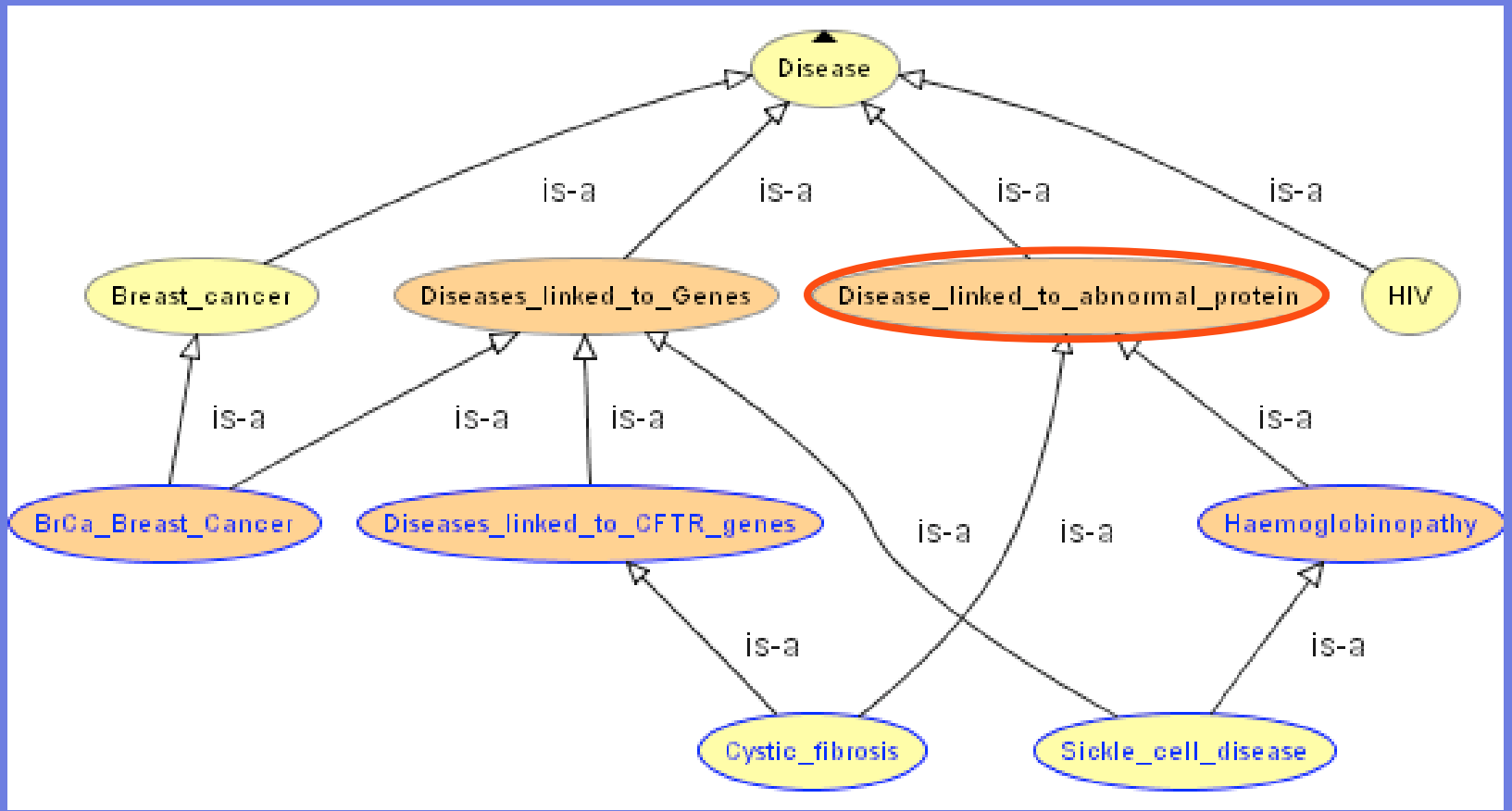
Let the classifier organise it



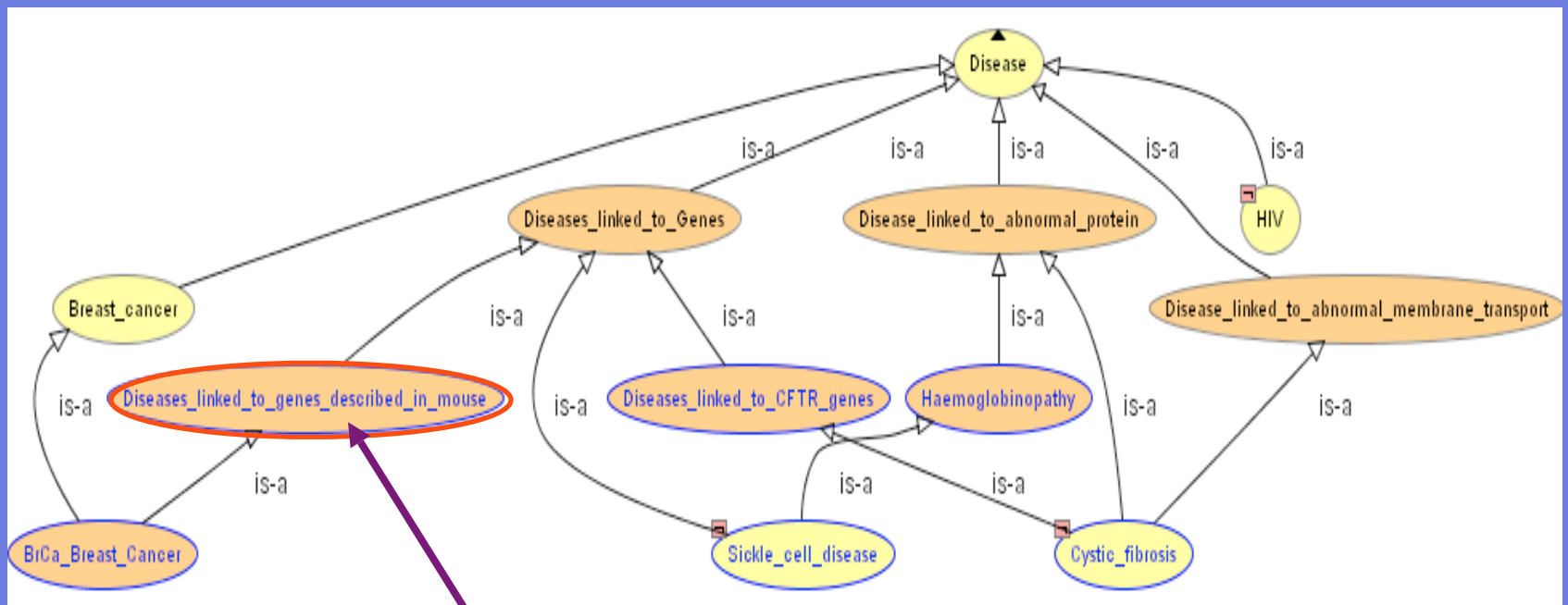
If you want more abstractions,



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 simply model for illustration only
  - Define a bacterial infection
  - Define a viral infection
  - Define a mixed (bacterial-viral) infection

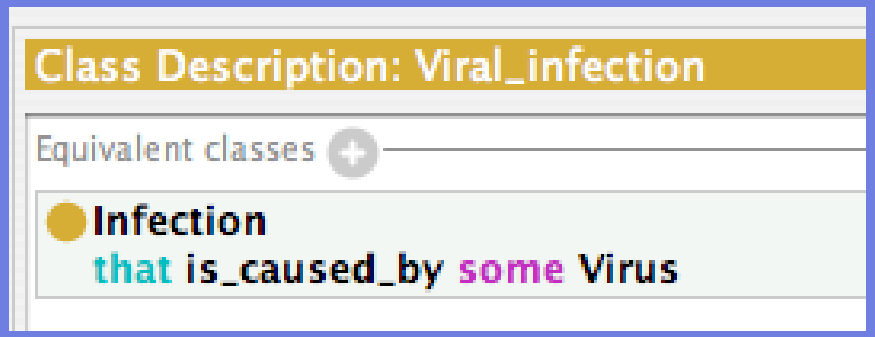
Bacterial infection

- *Any* infection caused by *some* bacterium

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

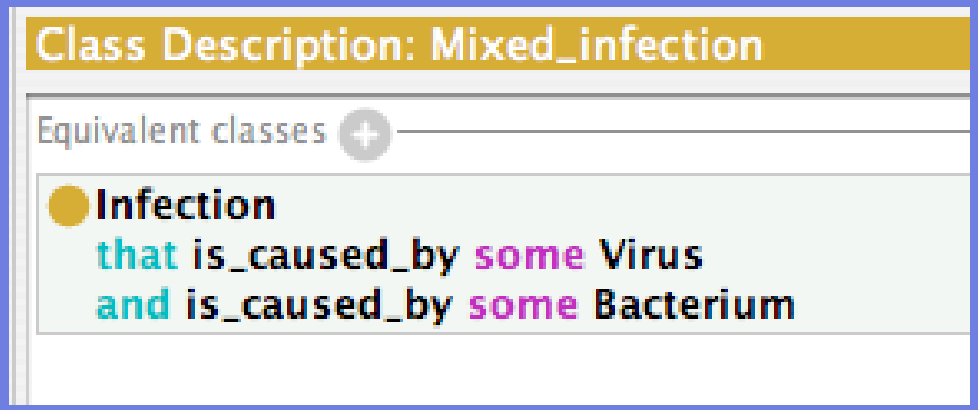
48

Viral infection



49

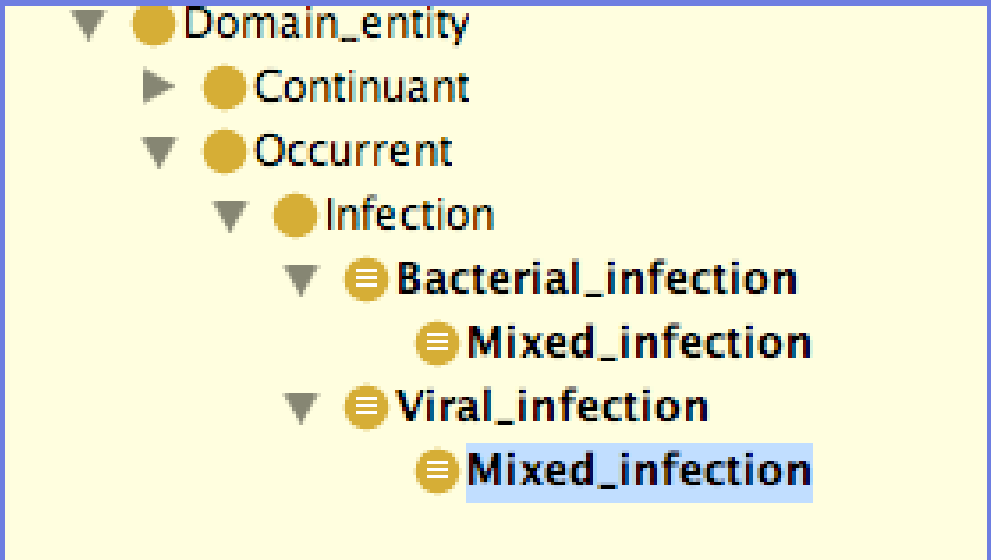
Mixed infection



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

50

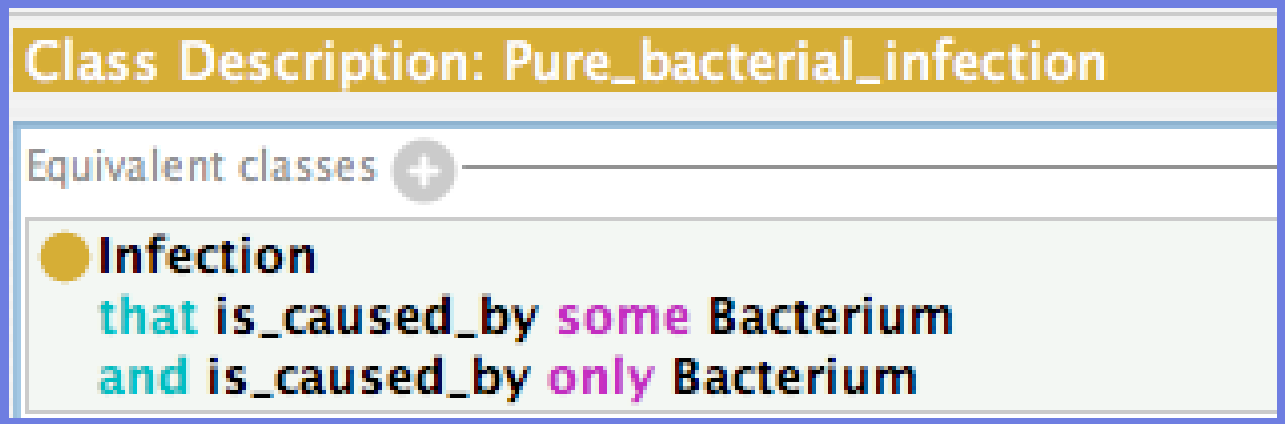
After classification



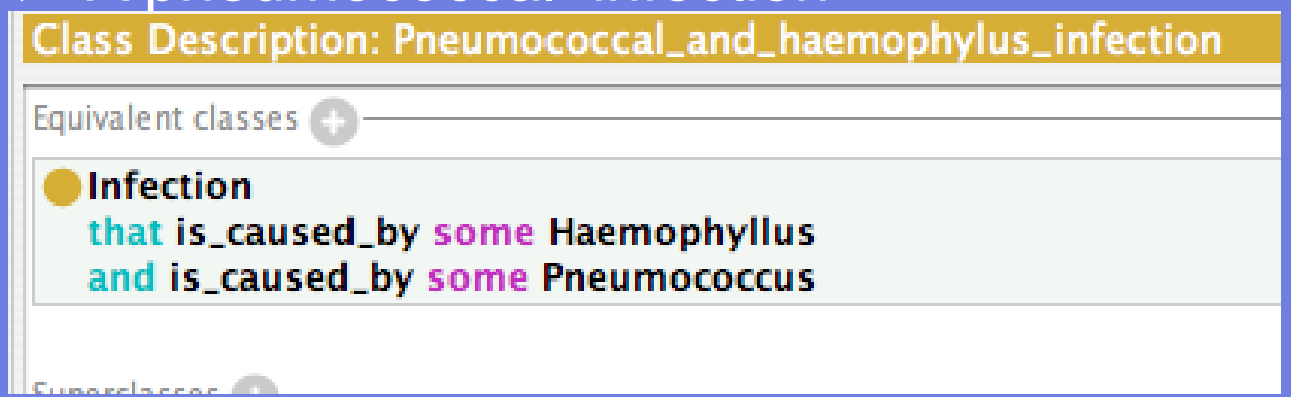
Why?

51

A “Pure bacterial infection”

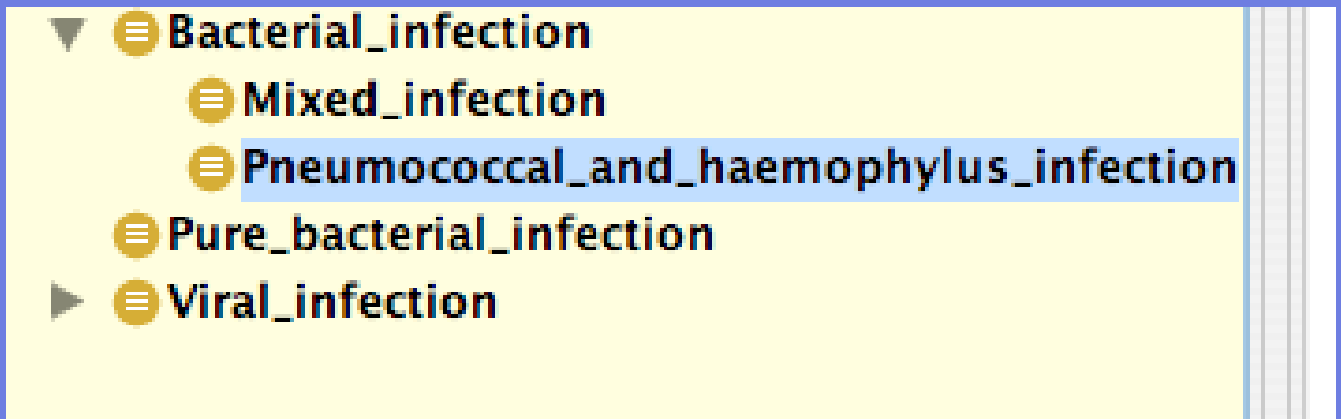


- A pneumococcal infection



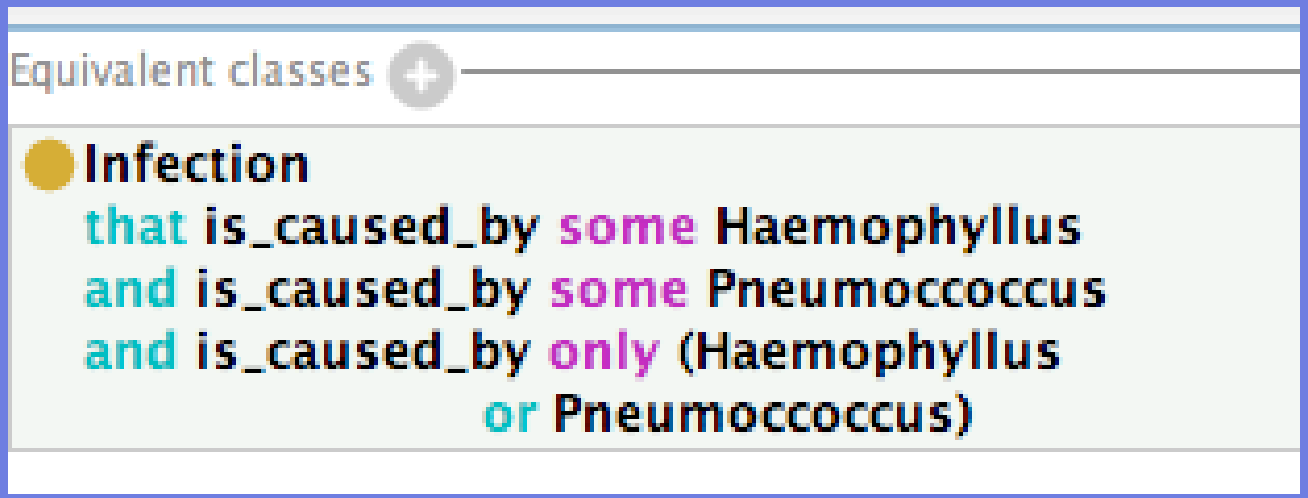
- How will these classify
  - Is a pneumococcal and haemophylus infection a kind of pure bacterial infection? - Both are

Why not?



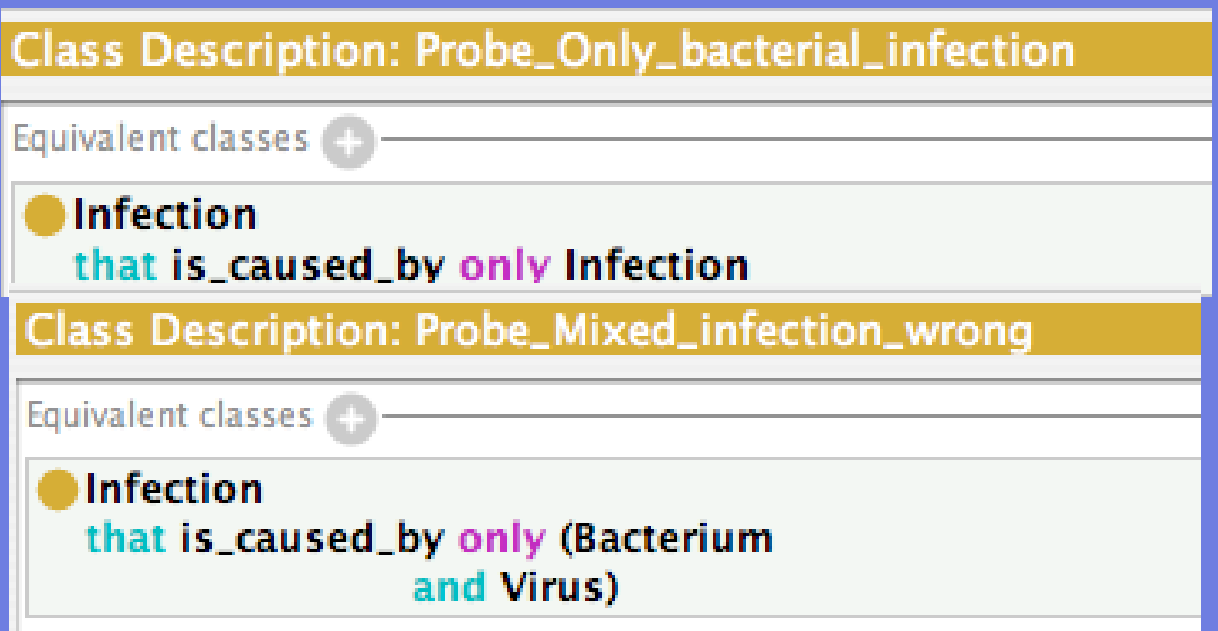
53

Closure Axioms



54

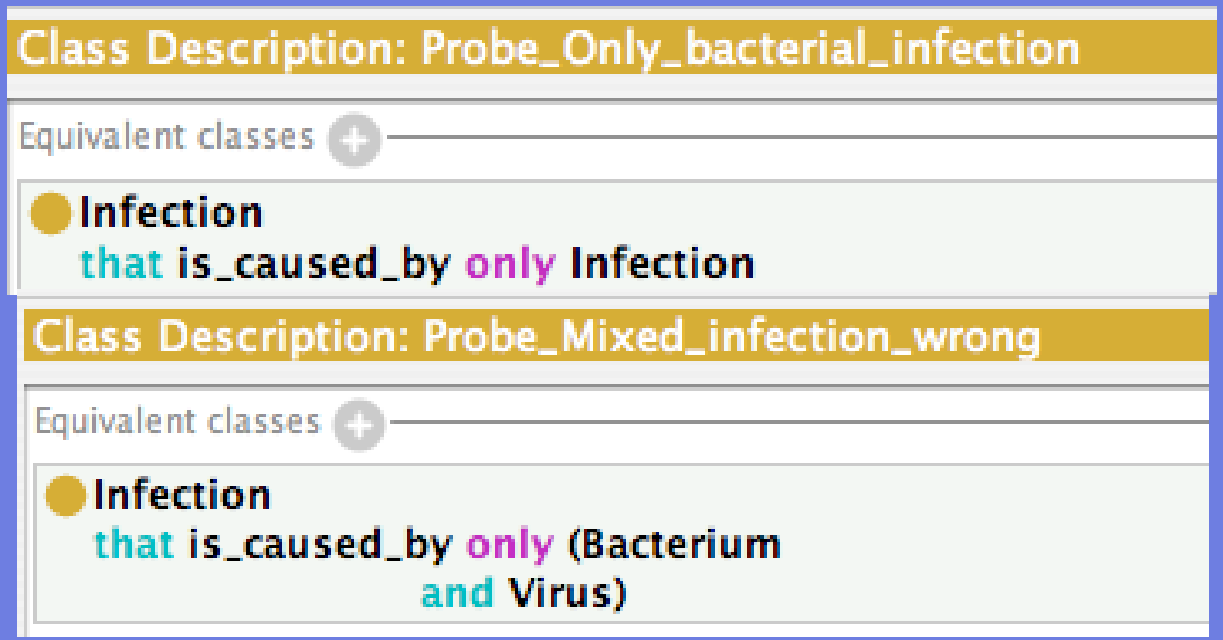
Trivial satisfiability
Two common errors

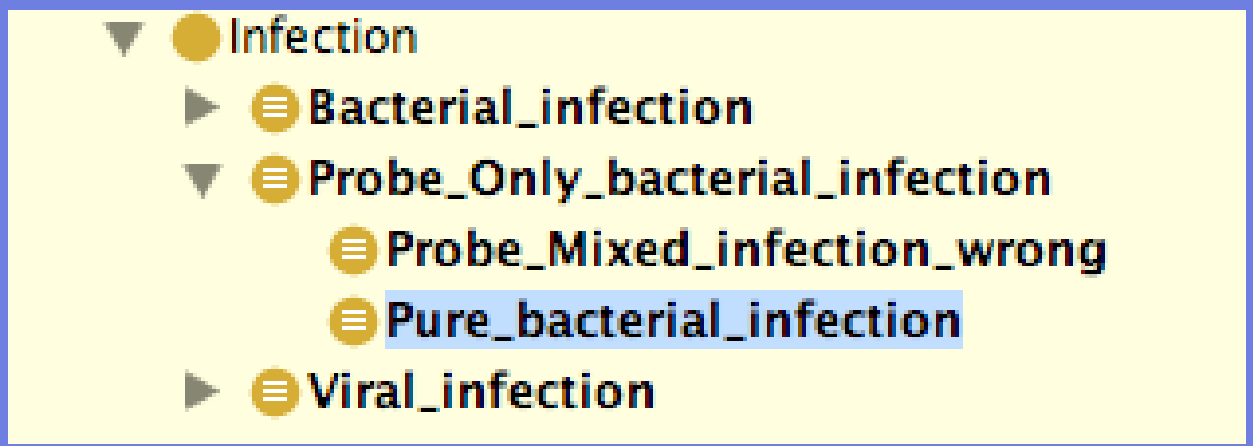


- How will these classify? Why?

55

After classification (Explain it)





56

Trivial satisfaction

- Bacterium and Virus are disjoint
  - Nothing is both a bacterium and a virus
    - owl:Nothing = (Bacterium AND Bottom)
- ONLY NOTHING ↔ 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*

58

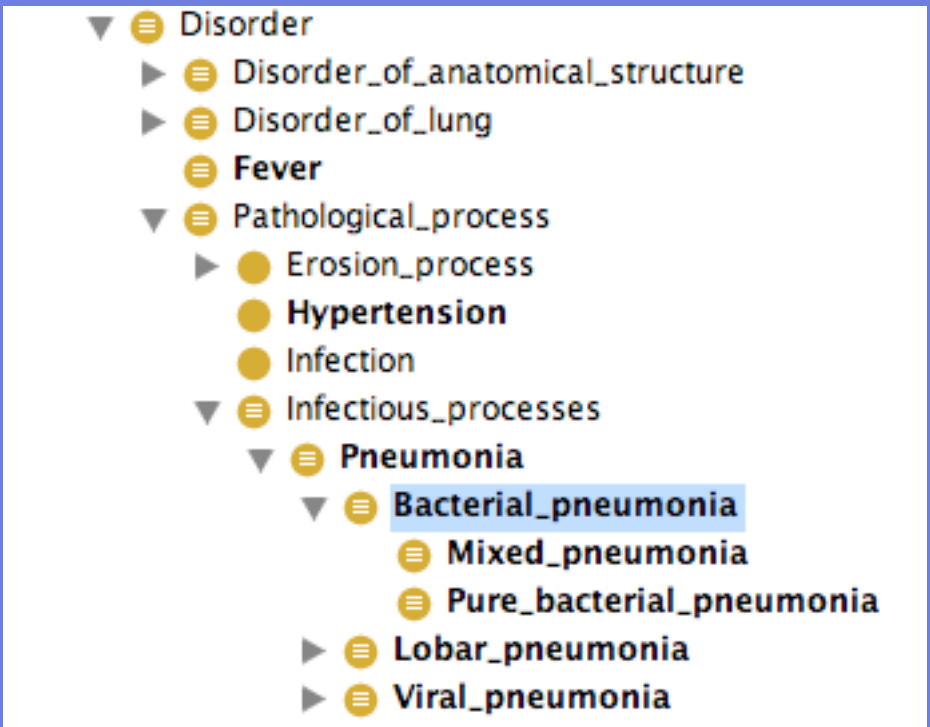
Why use modules

- 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.
      - *usualy 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



61

Protege-OWL import mechanism

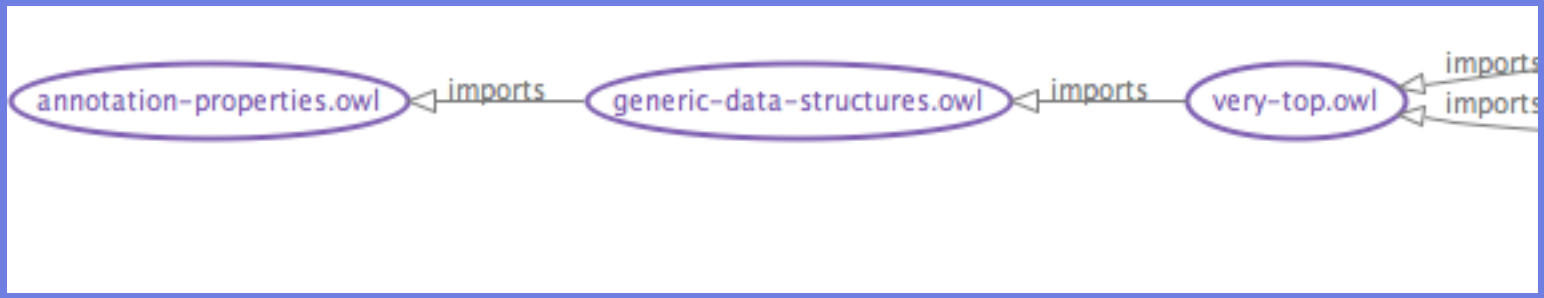
- Importer looks for a file with the correct identifying “Base URI”
  - Written into the header of the XML
  - NOT the physical location
- Order of search
  - (Specified file)
  - The local directory
  - Local libraries
  - Global libraries
  - The internet at the site indicated by the Base URI

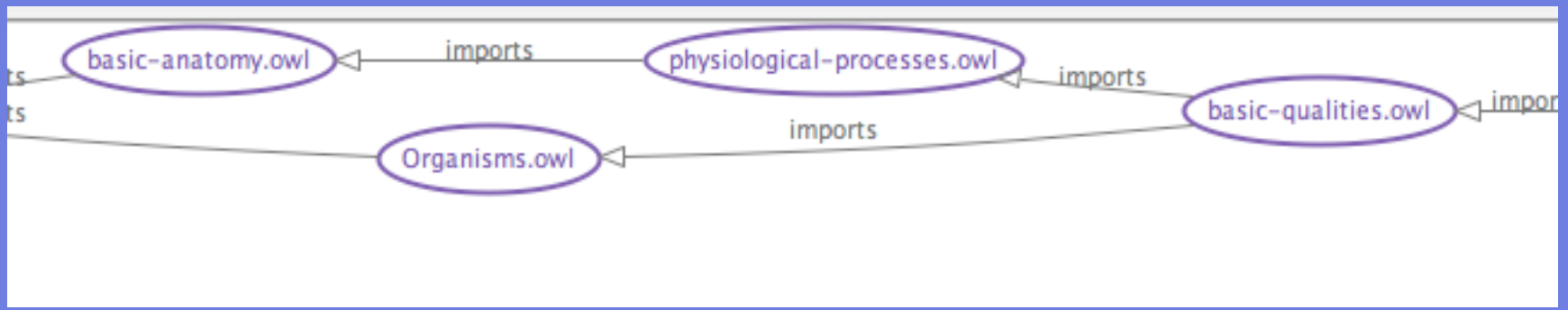
If you can, load the tutorial ontology nowontology/

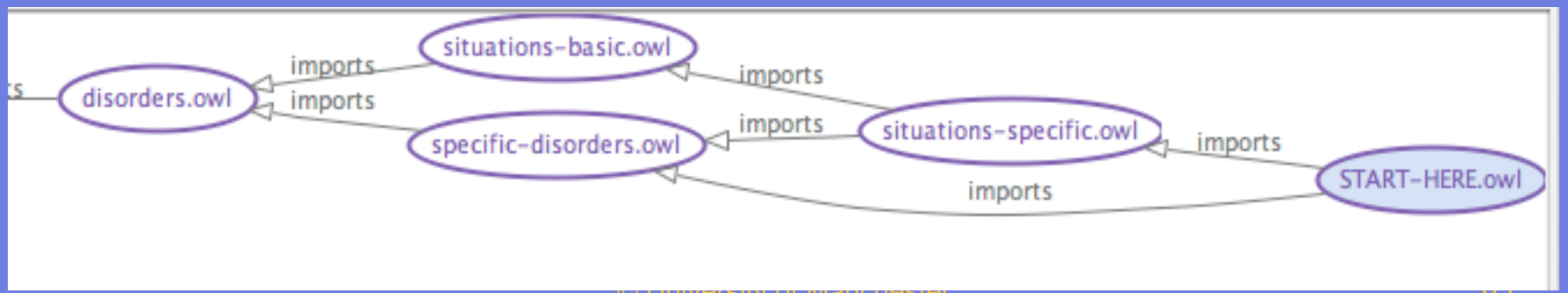
- Open .../biomedical-tutorial-2007/Start-Here.owl

Typical pattern

Views→Ontology views→OwlViz Imports







Module list

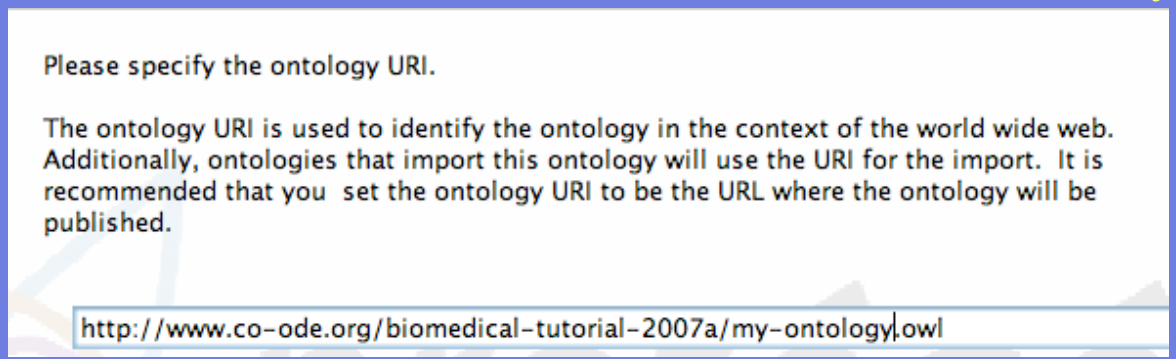
- Annotation - the annotation properties needed
- generic-data-structures -
  - quantities & numbers
- very-top - the upper ontology
  - in this case adapted for biomedicine
- Anatomy, Physiology, Biochemistry, Organism
  - The main topics
- Qualities
  - The basic qualities of those topics
- Disorders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

Anatomy and Disorders

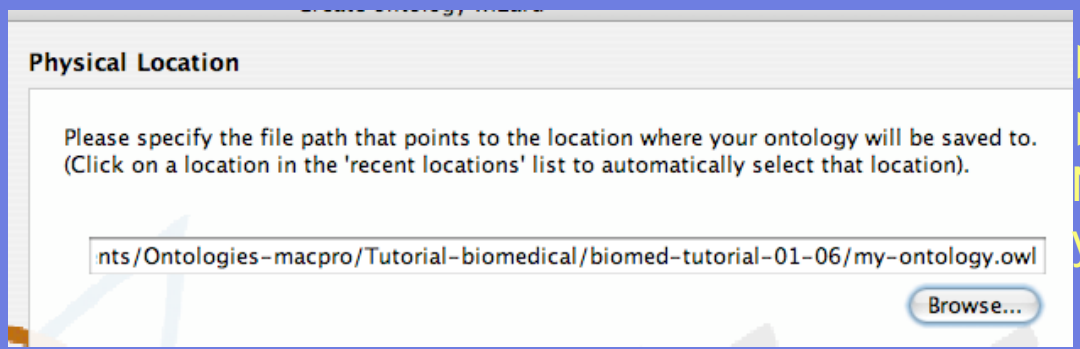
- Disorders have a locus in an anatomical structure of physiological process
  - Disorder has_locus SOME Anatomical_structure
- Disorders are anything which is described as pathological
  - has_normality_quality SOME Pathological
    - To be explained in detail late
- Parts and wholes
  - A whole field “mereology”
  - Multiple views - functional / cinician’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

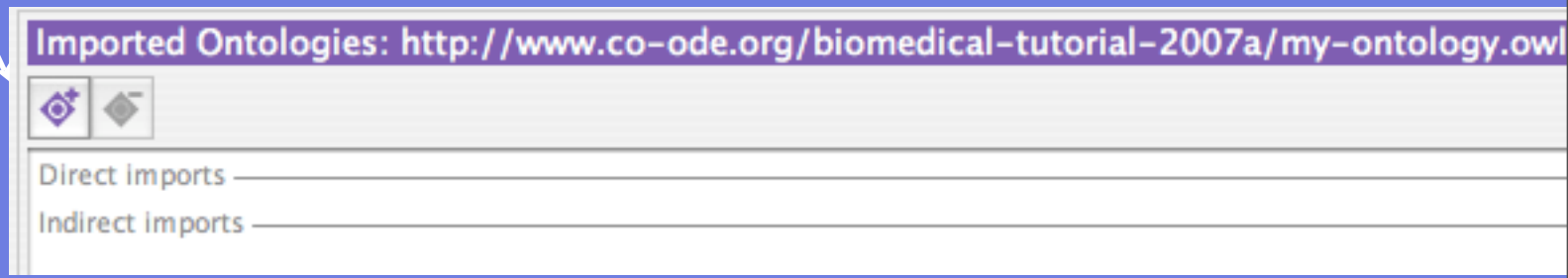


  - Press finish
  - NB This does NOT save your ontology!

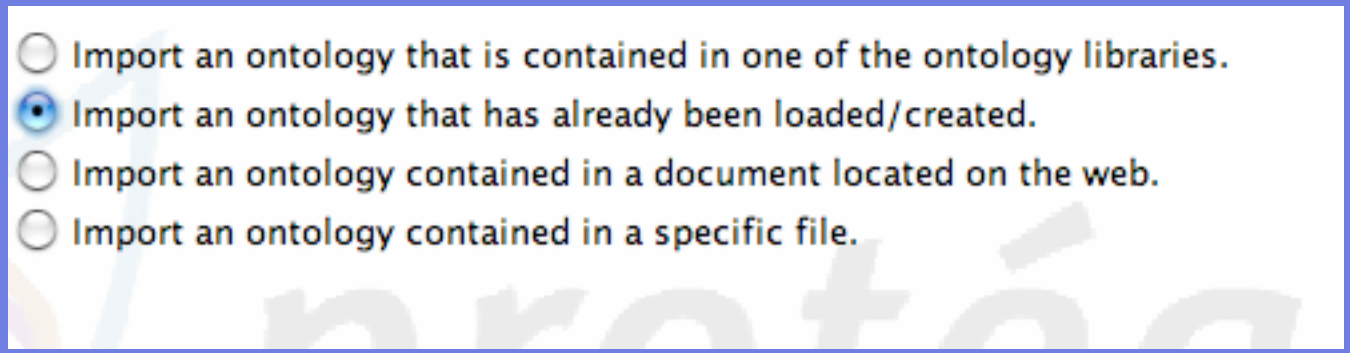
67

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

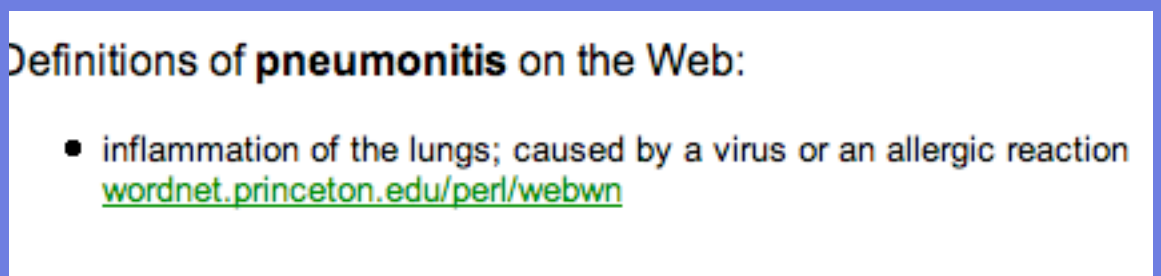


- Select disorders.owl and press finish

68

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

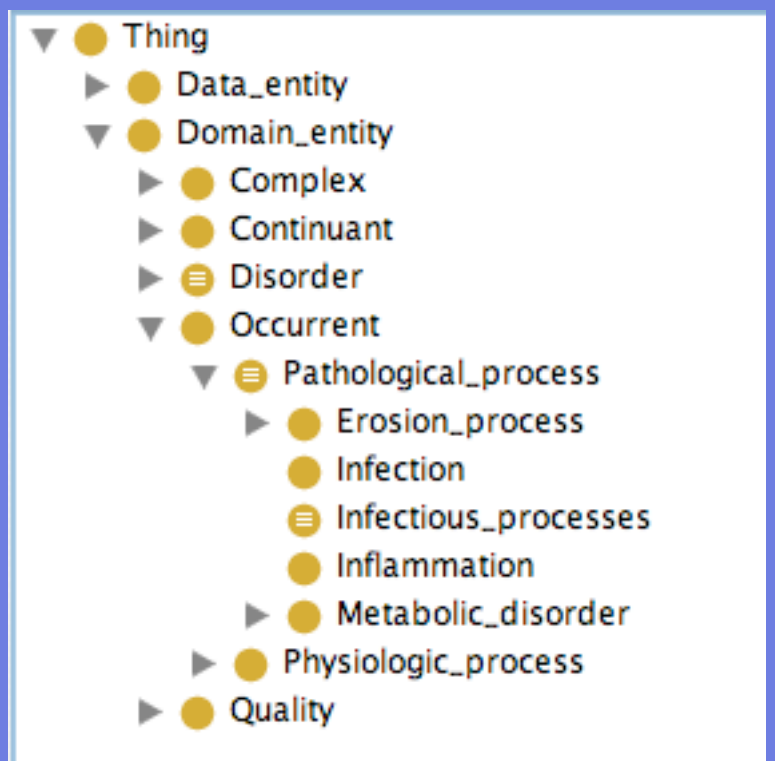


- Write your own paraphrases:
- “Pneumonitis” is an “Inflammation of the lungs”
- “Pneumonia” is an “inflammation of the lungs caused by an infection”
- Many definitions on the web, but this summarises them for our purposes.

69

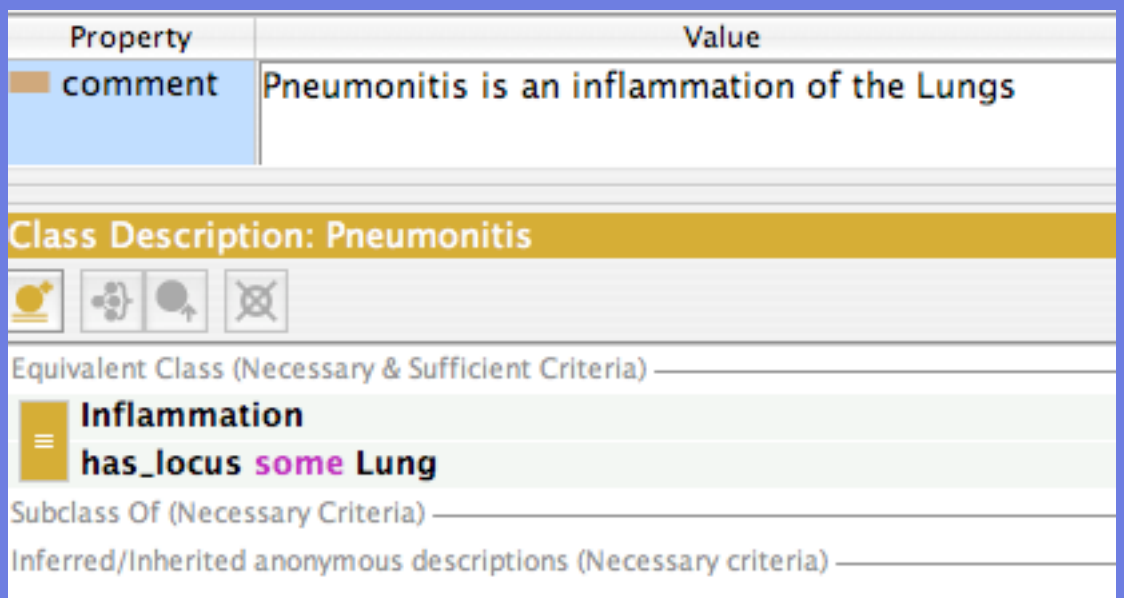
First defintion of pneumonitis

- "Inflammation of the lung"
  - Find Inflammation
    - CTRL or CMND F in class hierarchy



Create “Pneumonitis”

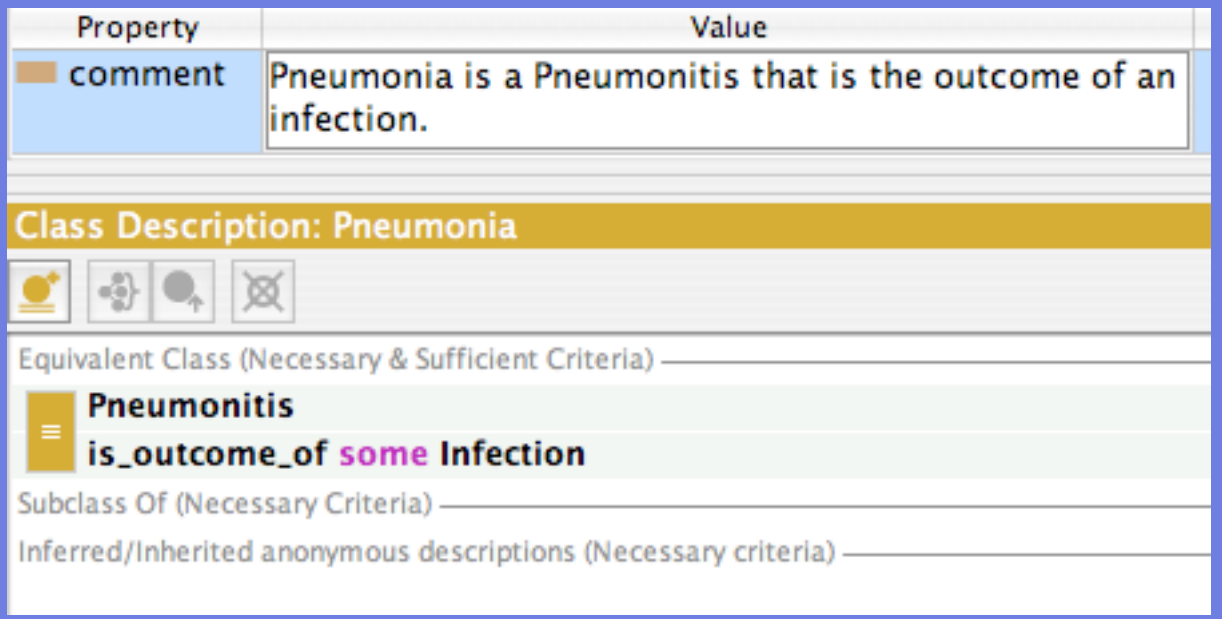
- Create a new subclass of Inflammation
  - In the comment box type something like “Pneumonitis” = “Inflammation of lung”
    - ALWAYS add a free text paraphrase of what you are modelling
  - Add the restriction
    - has_locus SOME Lung
  - Make it a defined class
    - CTRL/CMD-D.



71

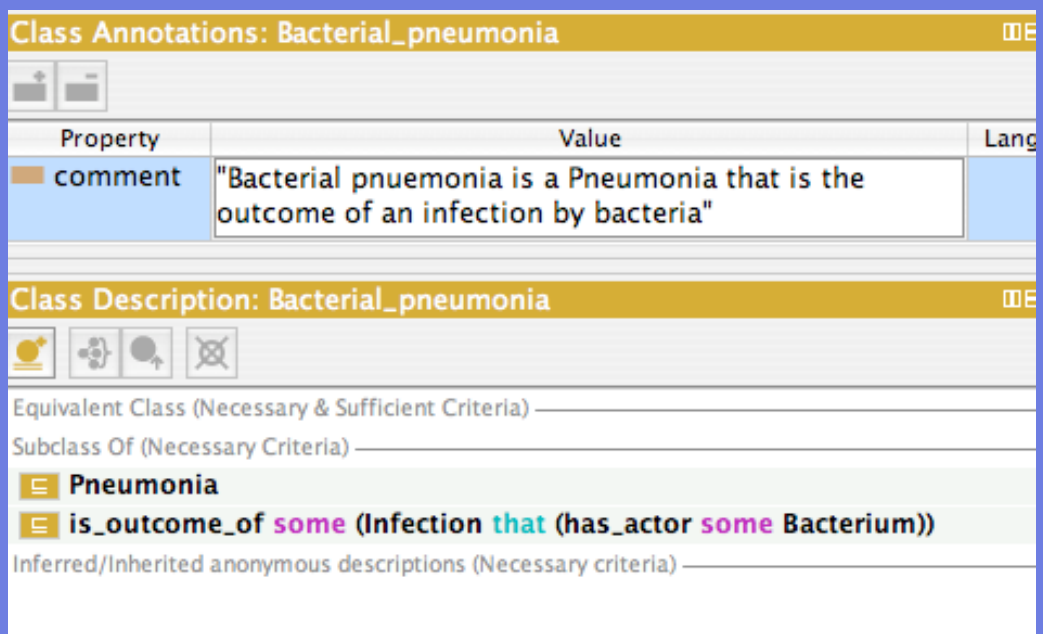
Create pneumonia

- "Pneumonia" is a pneumonitis is the outcome of an infection
  - In this ontology we use "is_outcome_of" for "cause"



Bacterial pneumonia

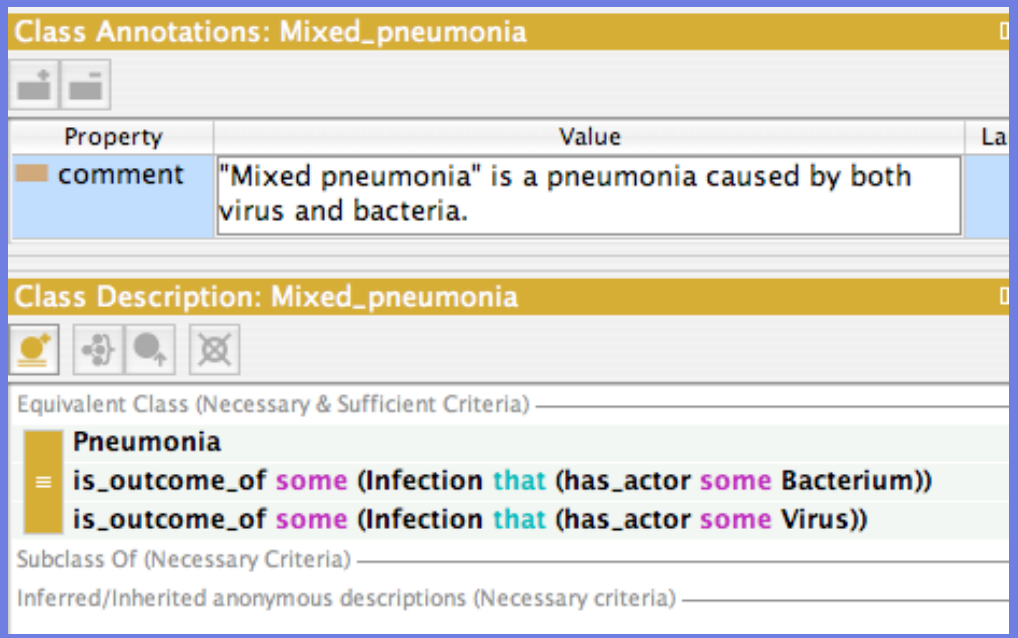
- First attempt
  - "Pneumonia caused by a bacteria"
- But need to rephrase to fit the ontology
  - "Pneumonia that is the outcome of an infection by bacteria"
    - In this ontology "by" translates to the property "has_actor"
      - *Processes have actors and objects*



73

By analogy make viral pnemonia and mixed pneumonia

- Mixed pneumonia is a pneumonia that is caused by both virus and pneumonia
  - How to say this



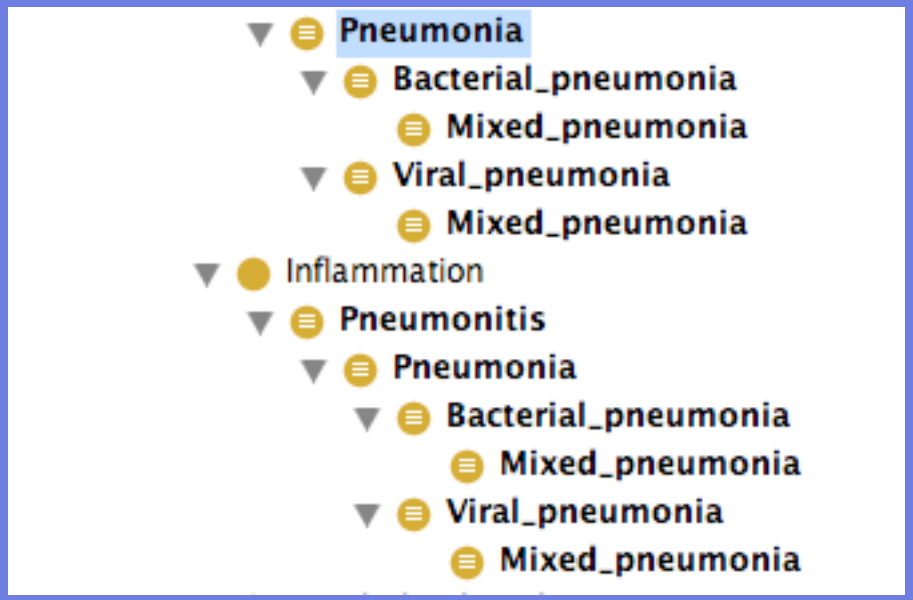
  - WARNING
    - wrong: has_actor SOME (Virus AND Bacterium)
      - *Nothing is both a virus and a bacteria*

74

Classify and check

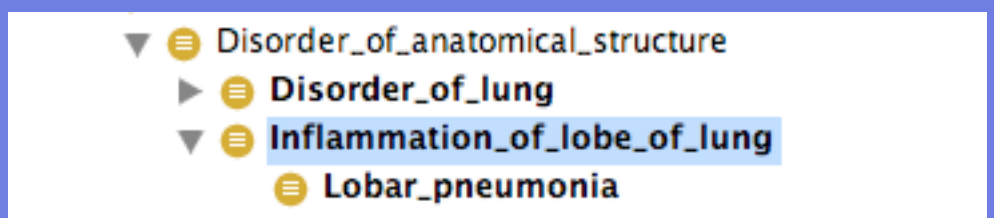
- Be sure that all classes are defined
  -  defined
  -  primitive
- To convert from primitive to defined, cmnd-d or ctrtl-d (Mac or PC)

Should get



What about “left lower lobe pneumonia”?

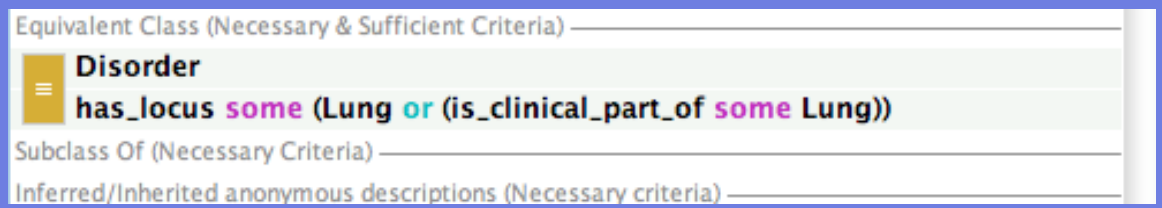
- First define lobar pneumonia as
  - Pneumonia that has locus in a lobe of a lung
    - “Lobe THAT is_subdivision_of SOME Lung”
- But what then is a disorder of the lung
  - Disorder THAT has_locus SOME Lung
- But what if I define an inflammation of a lobe of the lung
  - Inflammation THAT has_locus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

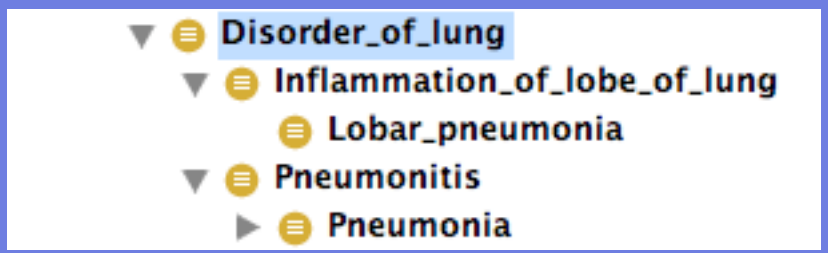


77

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)

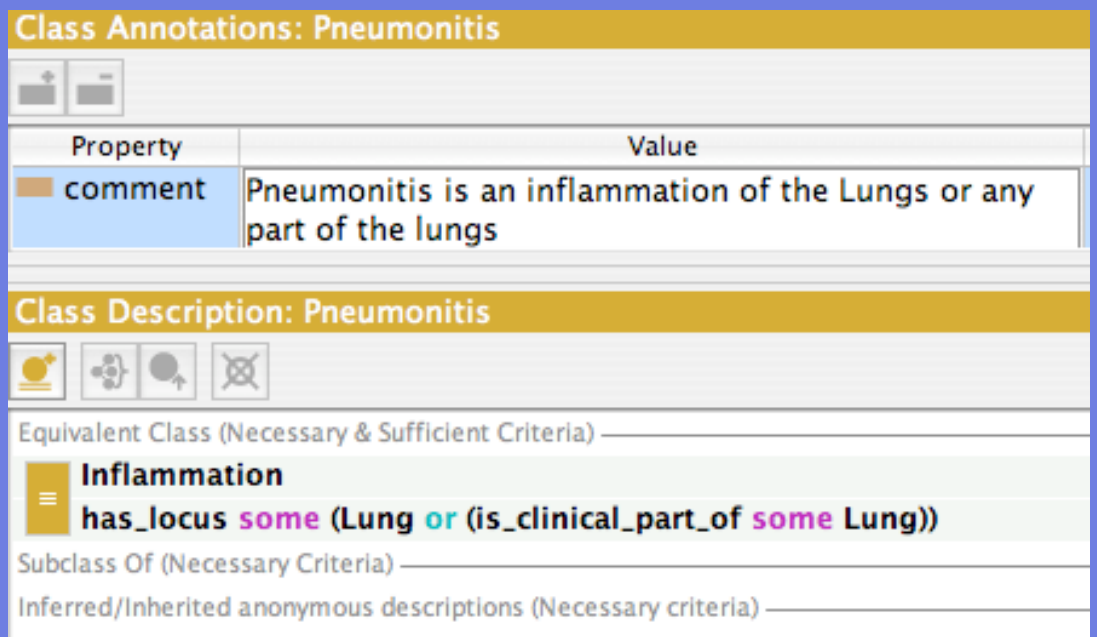




- Almost OK, but still Inflammation of lobe of lung is not a pneumonitis

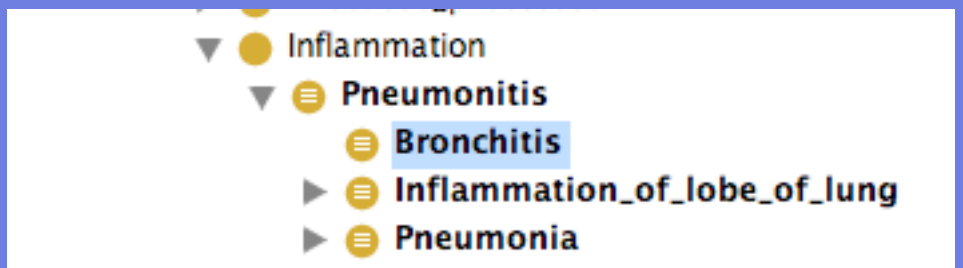
78

Make the pattern consistent

- Redefine Pneumonitis
  "An inflammation of the lung or any clinical part of the lung of the lung"
- 

Almost correct, but...

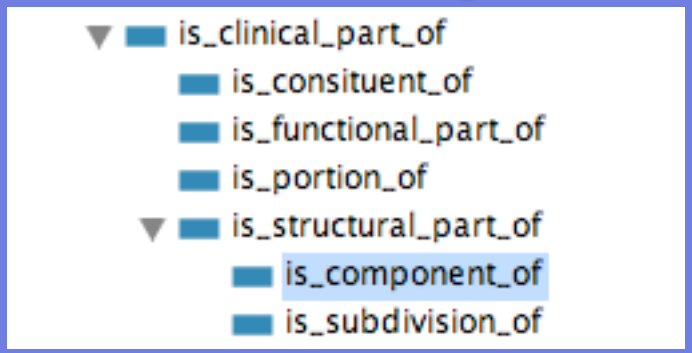
- What about “Bronchitis” ?
  - An inflammation of the bronchi (or any of their parts)
    - Try it and see.

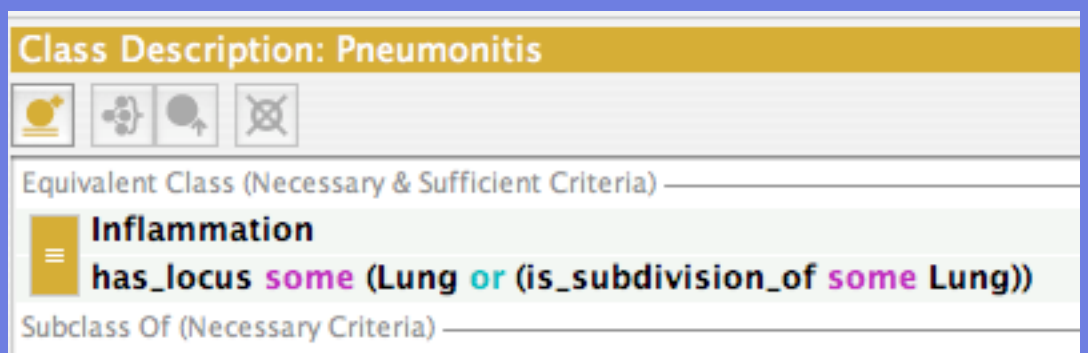


- Definition of “Pneumonitis” is now too broad
- Not just any part of the lung, but the “subdivisions” of the lung
- lobes, quadrants, bases, apices, etc.

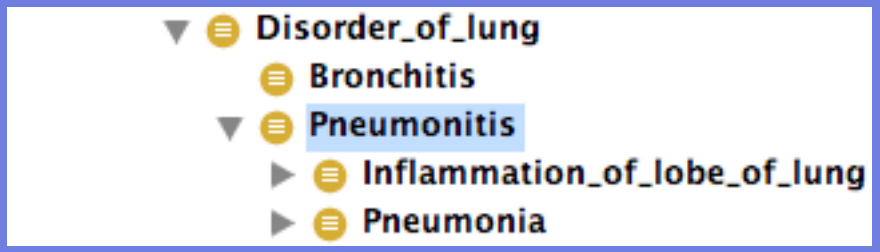
80

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
- 

Now reclassify

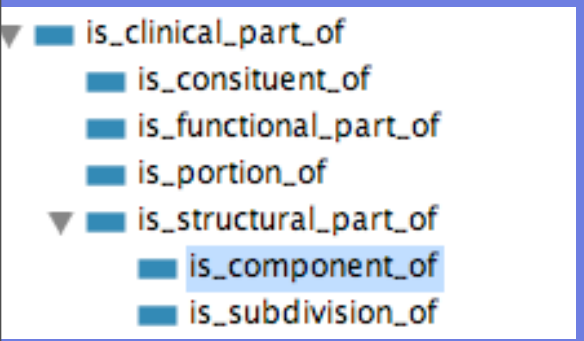


- Bronchitis is now a disorder of the lung (“lung disease”) but not a pneumonitis
  - As required.

82

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.

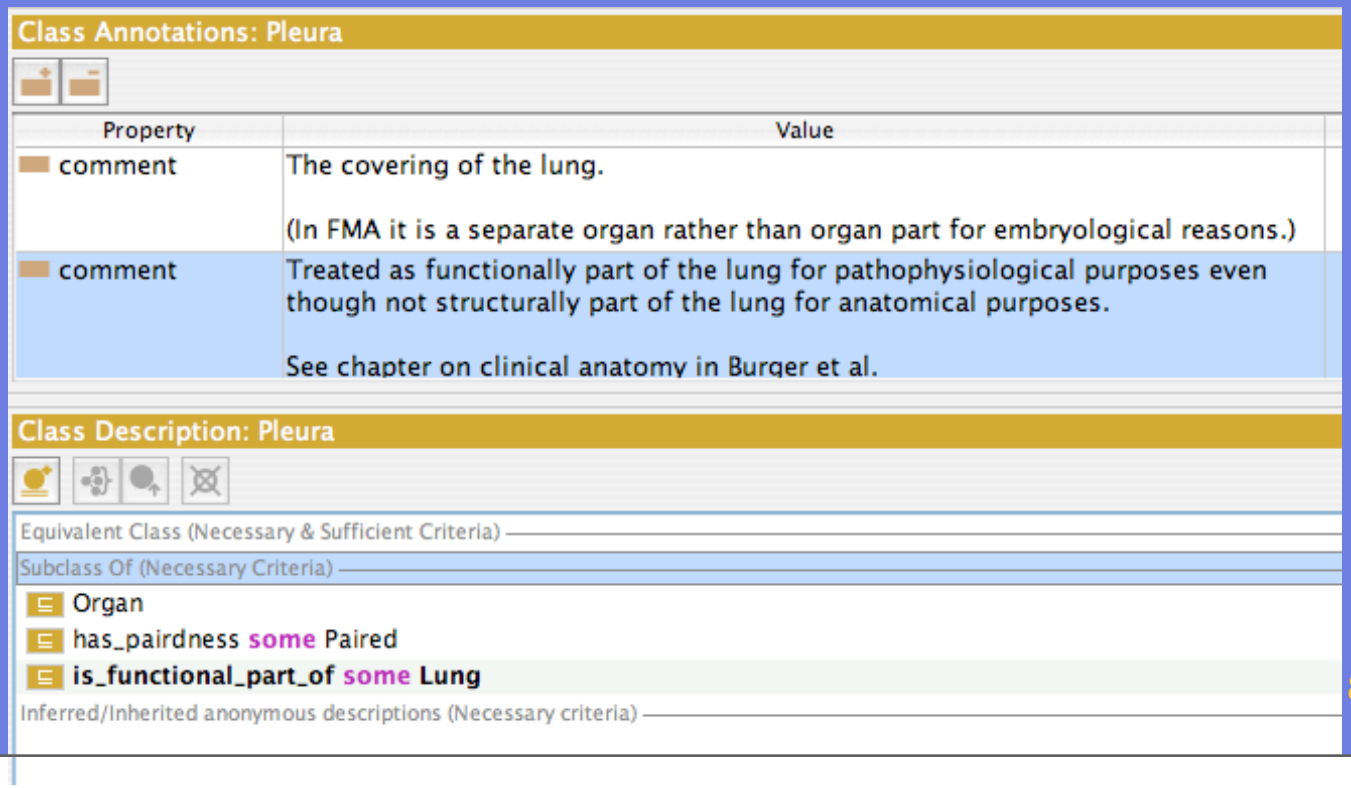


- Anything that is structurally a part of something is a clinical part of it
- Anything that is functionally a part of something is a clinical part of it
- etc.
- BUT NOT VICE VERSA.

Also affects modularity

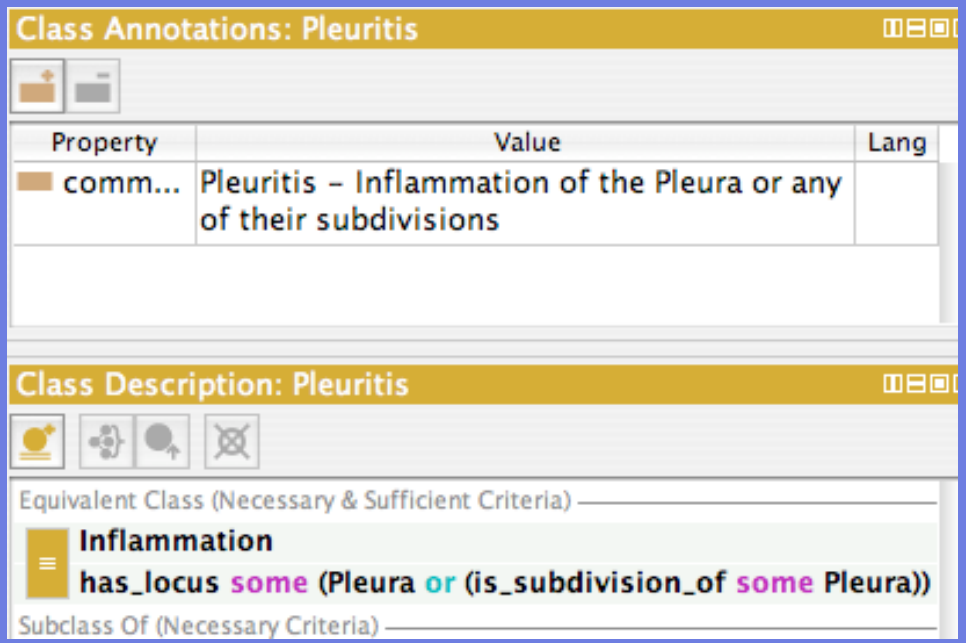
- We have chosen to model functional parts with physiology rather than with anatomy
  - To stick with the FMA view as far as possible in the Anatomy module.
- So we add the fact that the pleur are functionals part of the Lung in the physiologic_processes module rather than the anatomy module
  - Might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

Additions in *physiological_processes.owl*

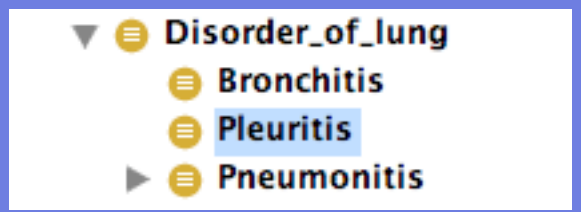


84

Create pleuritis and classify



- 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*



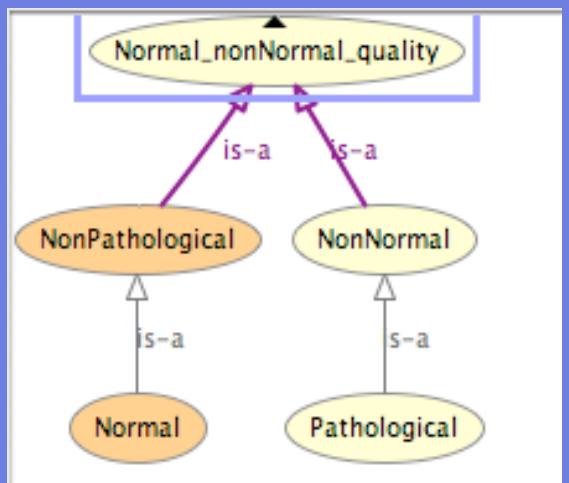
85

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

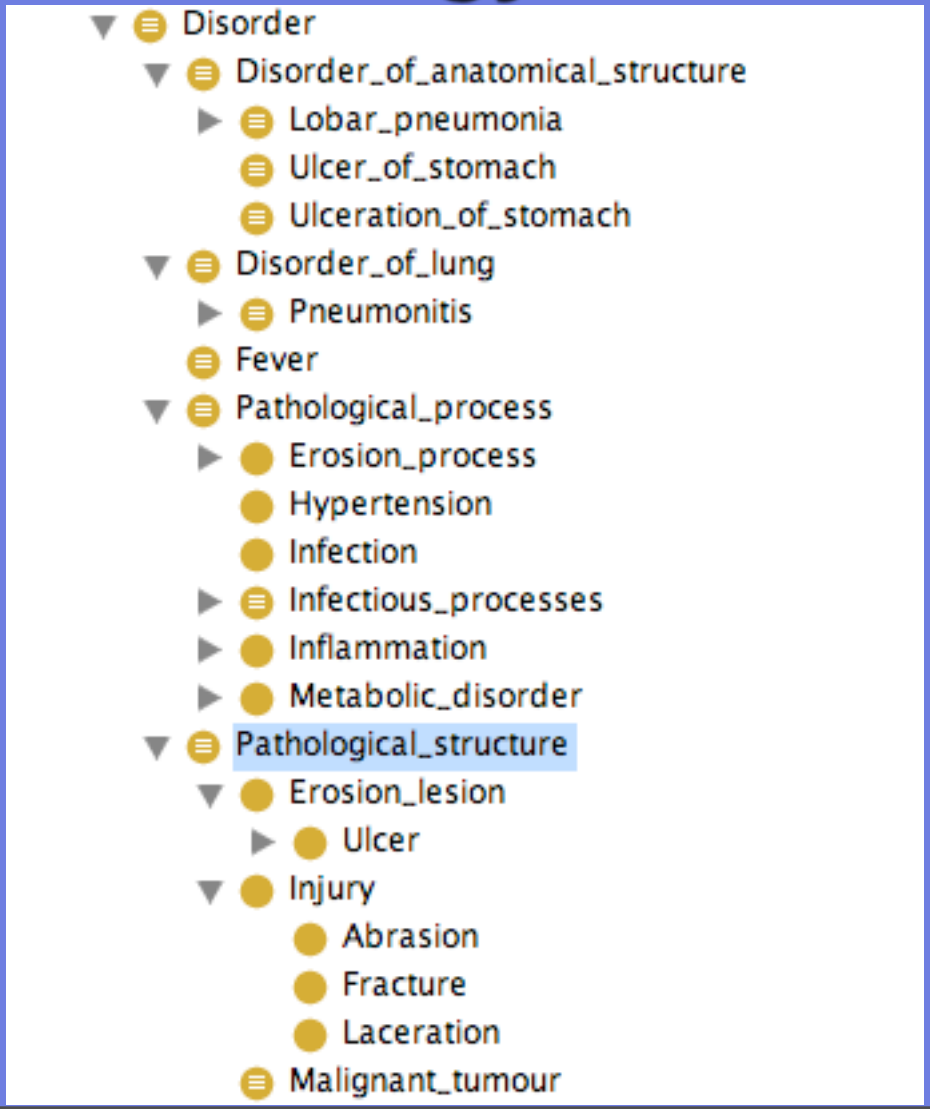
- Basic rules
  - Pathological ➞ nonNormal
  - Normal = NOT nonNormal
  - nonPathological = NOT pathological
- Remember
  - subclassOf means “necessarily implies”
    - so Pathological is a subclass of nonNormal
  - See the definitions of Normal-nonNormal_quality in disorders.owl
- Let the classifier 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



89

Note

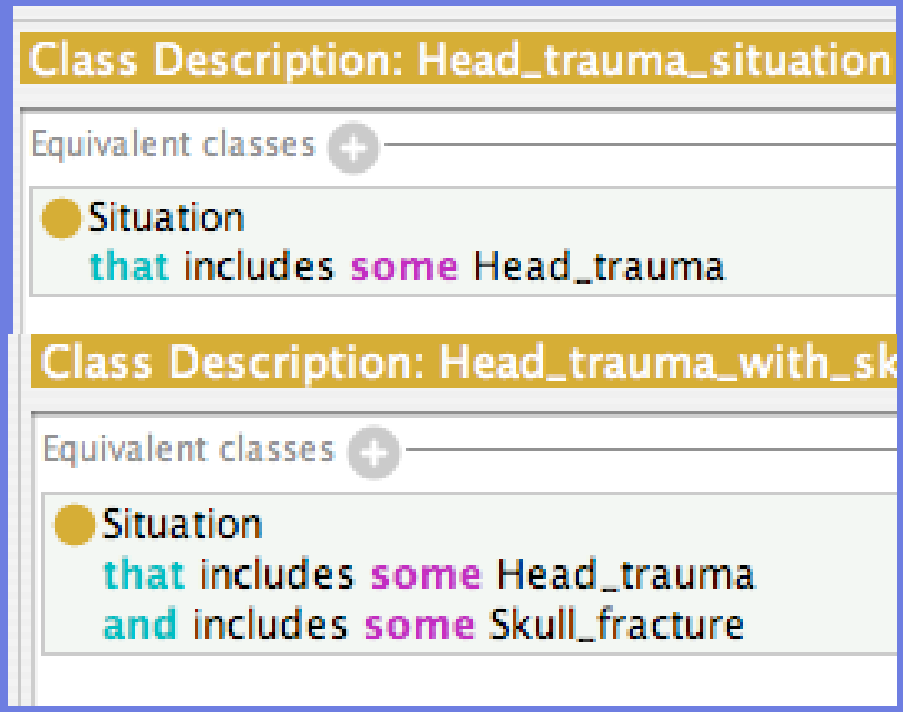
- Commented version of my_ontologies is in
  - Specific_disorders.owl
    - Make that the active ontology and compare

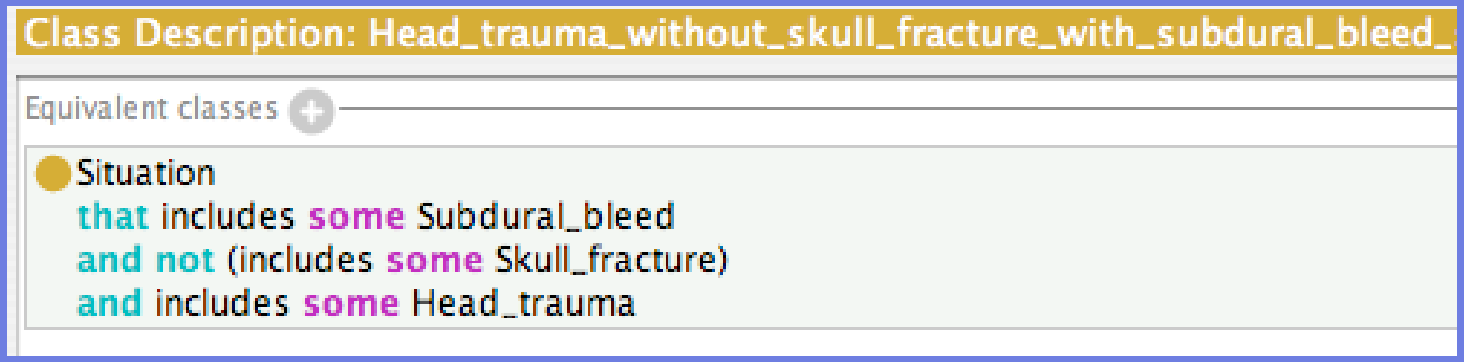
90

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 a event in a record is to assert that the event is a 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 classificaiton of results with recognition of equivalence

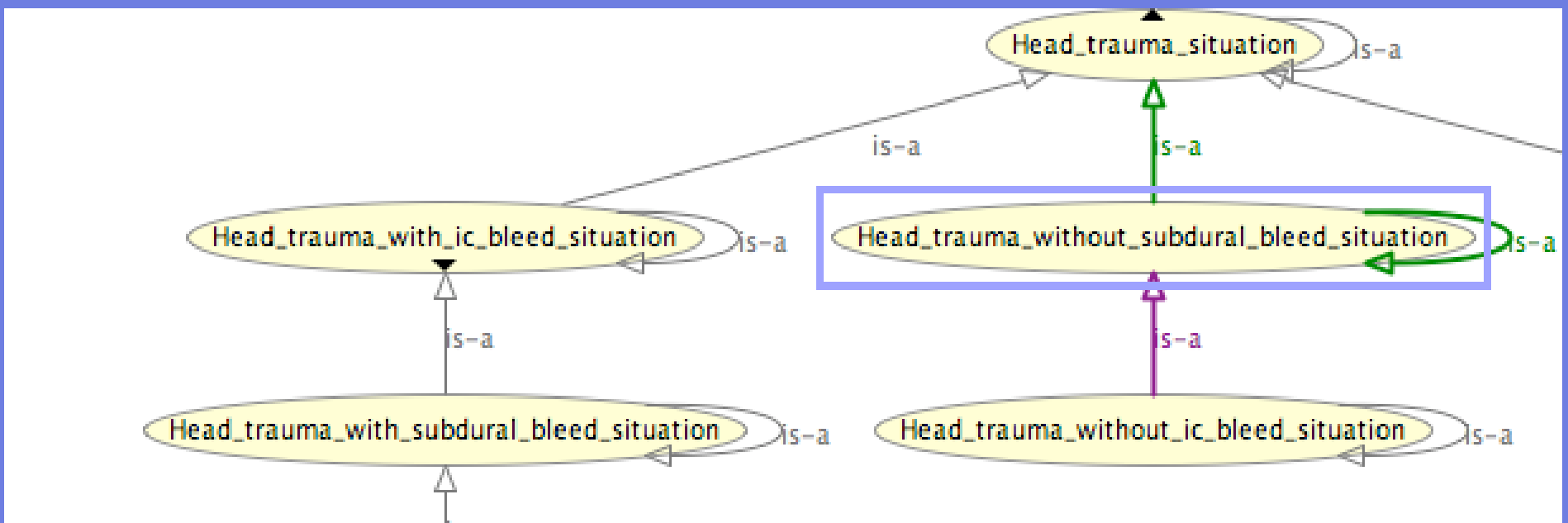
Example: Head Trauma with/without intracranial bleeding with/ without Skull Fracture





92

Classification - Note automatic inversion of negatives



93

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

94

What OWL (DL) cannot do

- A subset of first order logic with two variables
  - "Binary relational"
    - n-ary relations require "reifying" the relation
      - i.e.re-representing the property as a class
        - » See n-ary relations note at W3C SWBP
    - n-ary predicates can always be represented as binary predicates
      - subject to other limitations of OWL
  - No representation for "same"
    - The Owner is the same as the Person responsible
      - But often can manage this by property-chains
  - No representation for "All-All" statements
    - "All licensed drivers are authorised to drive all cars"
      - Known to be tractable but no implementation available
  - Representation of "optional" properties is not standardised. Options:
    - Min cardinality 0
    - Member of the domain
    - Other.
  - No representation of uncertainty
  - No representation of defaults and exceptions

95

What OWL (DL) cannot do

- Limited meta-representation
  - OWL DL is strictly first order
  - OWL-Full allows higher order representations but does not support reasoning
  - OWL 1.0 annotations, but seriously impoverished
    - (Odd interaction with RDF syntax issues)
  - OWL 1.1 has puns
  - See SWBP note: *Classes as Values*
- No higher order predicates
  - OWL-DL is strictly first order
    - “Members of endagered species”
- No closed world reasoning
  - Everything must be closed exlicitly
    - Take care when transforming from databases

96

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”***

97