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

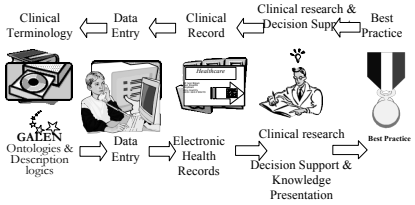
www.co-ode.org
www.clinical-escience.org
www.opengalen.org

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/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

2

Where I come from



© University of Manchester

"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*

8

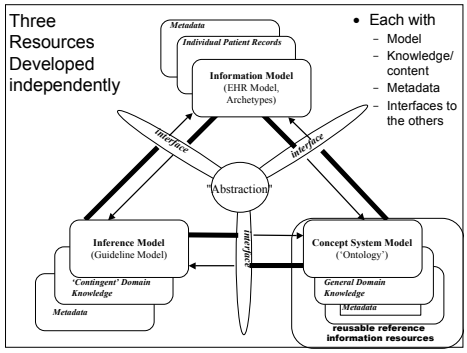
1

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

5

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
  - ...

2

•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
  - Open / closed world

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

© University of Manchester

MANCHESTER 1824

The University of Manchester

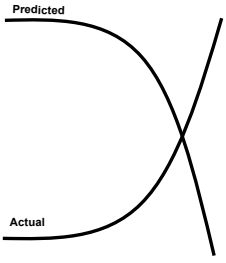
Fundamental problems:

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

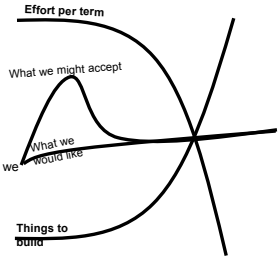


Combination of things to be done & time to do each thing

•Terms and forms needed
-Increases exponentially

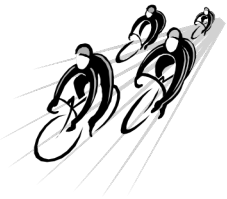
•Effort per term or form
-Must decrease to compensate

•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 ……



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

Defusing the exploding bicycle:

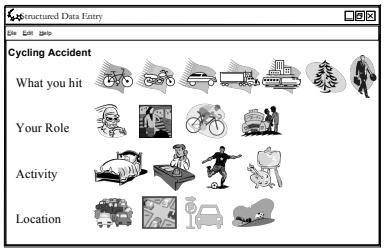
500 codes in pieces

- 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

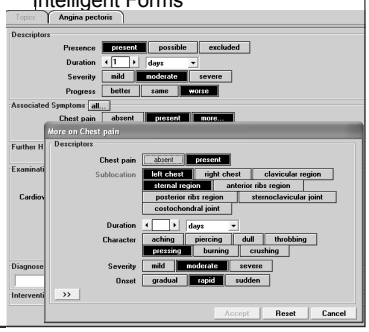
V12.24 Pedal cyclist injured in collision with two- or three-wheeled motor vehicle, unspecified pedal cyclist, nontraffic accident, while resting, sleeping, eating or engaging in other vital activities

Conceptual Lego… it could be...

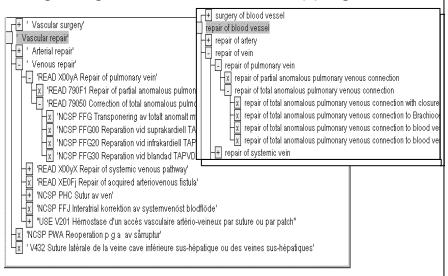
Goodbye to picking lists…



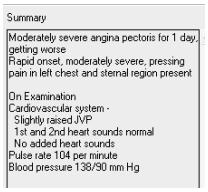
Intelligent Forms



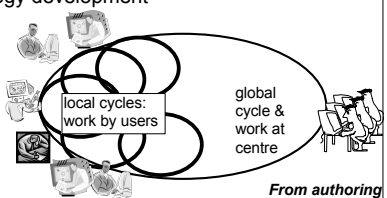
Integrating rather than Cross Mapping



And generate it in language



Supports Loosely coupled distributed ontology development

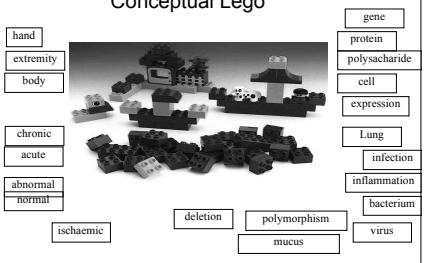


From 80% central/global effort to 10% central/global effort

From authoring to meta-authoring

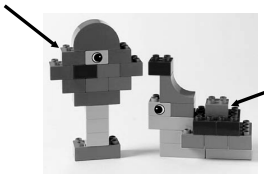
User effort cut by 75% compared with manual methods
Mostly in reduced committee meetings & arguments

The means: Logic as the clips for "Conceptual Lego"



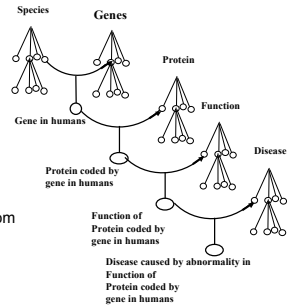
Logic as the clips for "Conceptual Lego"

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

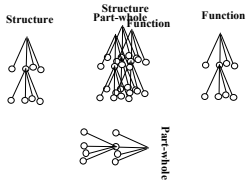


"***Hand*** which is ***anatomically normal***"

Build complex representations from modularised primitives



Normalising (untangling) Ontologies



© University of Manchester

7

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

© University of Manchester

Normalisation and Untangling

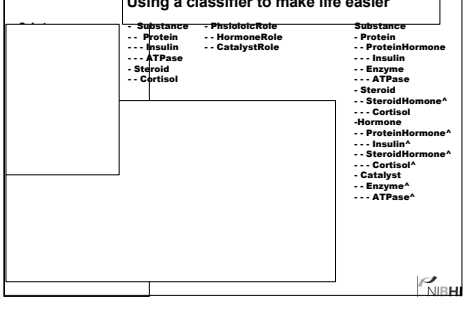
Let the reasoner do multiple classification

- 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

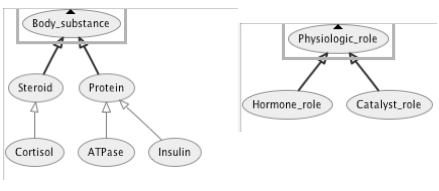
© University of Manchester

Untangling and Enrichment

Using a classifier to make life easier



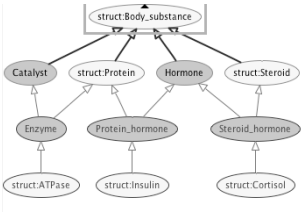
Modularised into structure and function ontologies (all primitive)



© University of Manchester

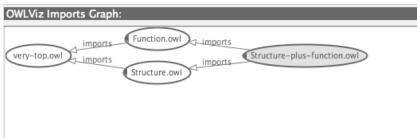
8

Unified ontology after classification



© University of Manchester

Module Structure



NB: Examples are in the tutorial ontology folder

© University of Manchester

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

© University of Manchester

Normalisation Criterion 2: *No hidden changes of meaning*

- *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

© University of Manchester

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…

© University of Manchester

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 & $\neg sub_1$ &…& $\neg sub_{n-1} \rightarrow sub_n$
- Heuristic:
  - Diagnosis by exclusion about self-standing concepts should *NOT* be part of 'standard' ontological reasoning

© University of Manchester

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 Parent & $\neg sub_1$ &…& $\neg sub_{n-1} \rightarrow sub_n$
- 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)

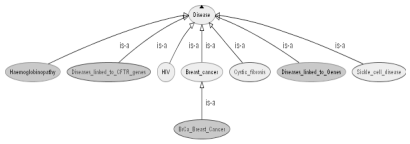
© University of Manchester

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*

© University of Manchester

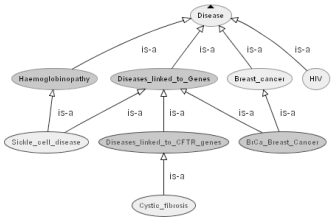
A real example: Build a simple treee



easy to maintain

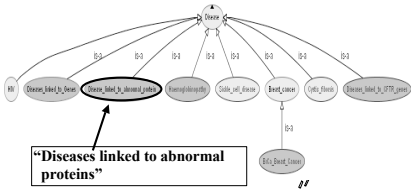
© University of Manchester

Let the classifier organise it



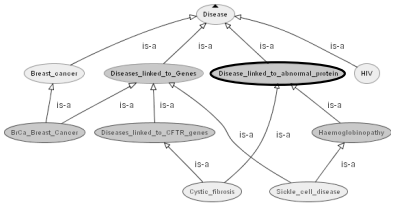
© University of Manchester

If you want more abstractions, just add new definitions
(re-use existing data)



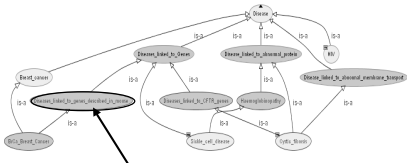
© University of Manchester

And let the classifier work again



© University of Manchester

And again – even for a quite different category



© University of Manchester

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*

© University of Manchester

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

© University of Manchester 47

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*

© University of Manchester 48

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 pneumoccocus is a 'pneumococcal pneumonia'*

© University of Manchester

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

© University of Manchester

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)

© University of Manchester

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"

© University of Manchester

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

© University of Manchester

Basic steps in building an ontology:1 Gather your terms

Lung
Virus
Haemophylus
Pneumonitis
Bacteria
Pneumococcus
Rales
Pneumonia
Antibiotic
Chest

© University of Manchester

Basic steps in building an ontology 2: Organise your terms

Lung
Pneumonitis
Chest
Pneumonia
Micro-organism
Virus
Bacteria
Rales
Pneumococcus
Haemophylus
Antibiotic

© University of Manchester

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

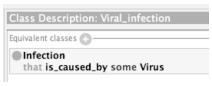
© University of Manchester

Bacterial infection

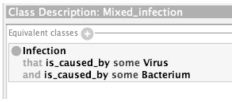
▶ *Any* infection caused by *some* bacterium

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

Viral infection

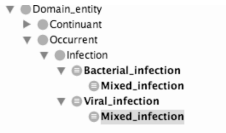


Mixed infection



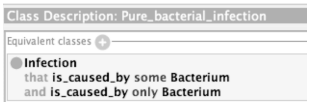
▶ How will bacterial infection, viral infection, and mixed infection classify?
- ▶ Run the classifier and see

After classification

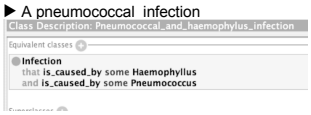


Why?

A "Pure bacterial infection"



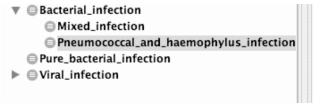
▶ A pneumococcal infection



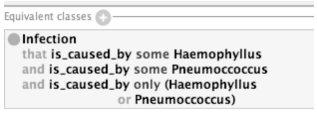
▶How will these classify

▶Is a pneumococcal and haemophylus infection a kind of pure bacterial infection? - Both are bacteria

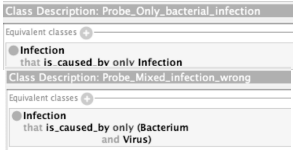
Why not?



Closure Axioms

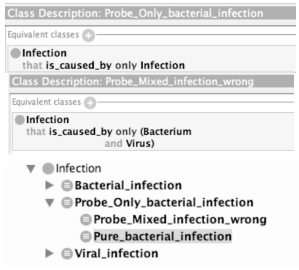


Trivial satisfiability
Two common errors



▶ How will these classify? Why?

After classification (Explain it)



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"

© University of Manchester

Modularisation: *towards assembling ontologies from reusable fragments*

© University of Manchester

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
    - Easy to put something in the "wrong" module

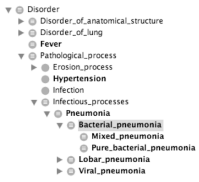
© University of Manchester

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

© University of Manchester

Items from active ontology are in bold



© University of Manchester

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

© University of Manchester

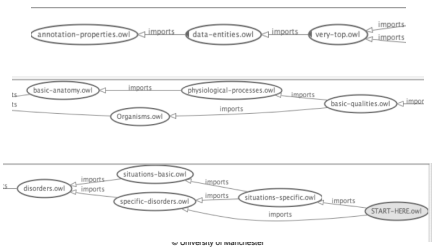
If you can, load the tutorial ontology nowontology/

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

© University of Manchester

Typical pattern

Views→Ontology views→OwlViz Imports



© University of Manchester

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

© University of Manchester

Anatomy and Disorders

A more realistic version of Pneumonias

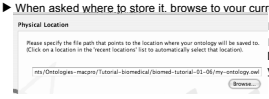
- Disorders have a locus in an anatomical structure of physiological process
  - Disorder has_locus SOME Anatomical_structure
- Disorders are anything which is described as pathological
  - has_normality_quality SOME Pathological
    - To be explained in detail late
- Parts and wholes
  - A whole field "mereology"
  - Multiple views - functional / cinician's view different from structural / anatomist's view.

© University of Manchester

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 "mv-ontology.owl"

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



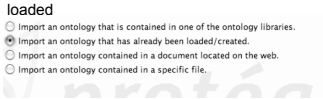
  - Press finish
  - NB This does NOT save your ontology!

19

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

© University of Manchester

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:

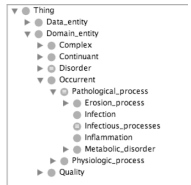
- inflammation of the lungs; caused by a virus or an allergic reaction
  wordnet.princeton.edu/perl/webwn

- 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.

© University of Manchester

First defintion of pneumonitis

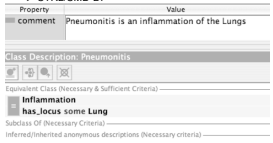
- "Inflammation of the lung"
  - Find Inflammation
    - CTRL or CMND F in class hierarchy



© University of Manchester

Create "Pneumonitis"

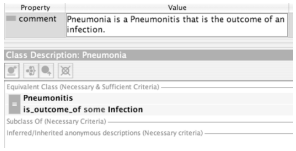
- Create a new subclass of Inflammation
  - In the comment box type something like "Pneumonitis" = "Inflammation of lung"
    - ALWAYS add a free text paraphrase of what you are modelling
  - Add the restriction
    - has_locus SOME Lung
  - Make it a defined class
    - CTRL/CMD-D.



20

Create pneumonia

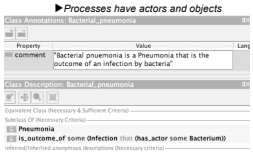
- "Pneumonia" is a pneumonitis is the outcome of an infection
  - In this ontology we use "is_outcome_of" for "cause"



© University of Manchester

Bacterial pneumonia

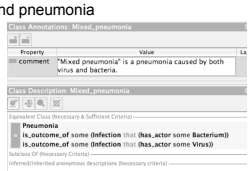
- First attempt
  - "Pneumonia caused by a bacteria"
- But need to rephrase to fit the ontology
  - "Pneumonia that is the outcome of an infection by bacteria"
    - In this ontology "by" translates to the property "has_actor"
      - *Processes have actors and objects*



© University of Manchester

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*

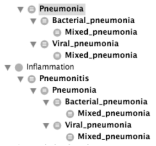
© University of Manchester

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)

© University of Manchester

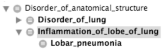
Should get



© University of Manchester

What about "left lower lobe pneumonia"?

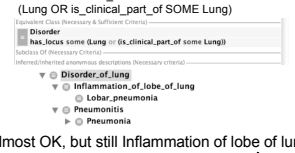
- First define lobar pneumonia as
  - Pneumonia that has locus in a lobe of a lung
    - "Lobe THAT is_subdivision_of SOME Lung"
- But what then is a disorder of the lung
  - Disorder THAT has_locus SOME Lung
- But what if I define an inflammation of a lobe of the lung
  - Inflammation THAT has_locus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



© University of Manchester

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)

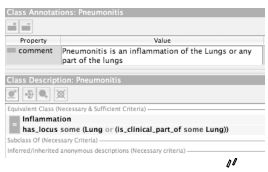


- Almost OK, but still Inflammation of lobe of lung is not a pneumonitis

© University of Manchester

Make the pattern consistent

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



© University of Manchester

Almost correct, but...

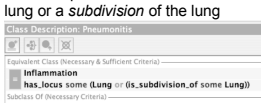
- Disorder_of_lung
  - Pneumonitis
    - Inflammation_of_lobe_of_lung
    - Pneumonia

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

© University of Manchester

The property hierarchy allows multiple views

- is_clinical_part_of
  - is_consituent_of
  - is_functional_part_of
  - is_portion_of
  - is_structural_part_of
    - is_component_of
    - is_subdivision_of

- 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

- Disorder_of_lung
  - Bronchitis
  - Pneumonitis
    - Inflammation_of_lobe_of_lung
    - Pneumonia

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

© University of Manchester

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_consituent_of
  - is_functional_part_of
  - is_portion_of
  - is_structural_part_of
    - is_component_of
    - is_subdivision_of

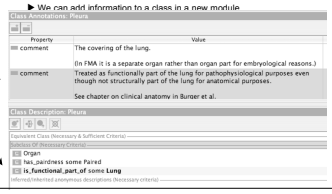
- Anything that is structurally a part of something is a clinical part of it
- Anything that is functionally a part of something is a clinical part of it
- etc.
- BUT NOT VICE VERSA.

© University of Manchester

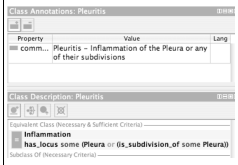
Also affects modularity

- 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 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*



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*

- Disorder_of_lung
  - Bronchitis
  - Pleuritis
  - Pneumonitis

© University of Manchester

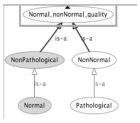
Normality and Negation

- 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

© University of Manchester

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...



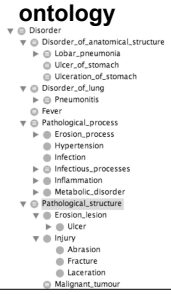
© University of Manchester

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***

© University of Manchester

Example from tiny tutorial ontology



Note

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

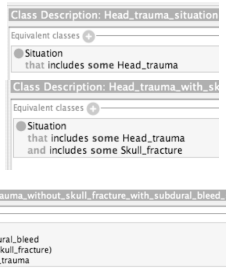
© University of Manchester

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

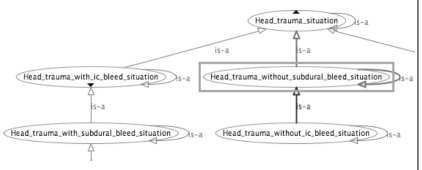
© University of Manchester

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



© University of Manchester

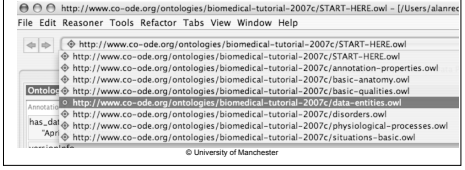
Classification - Note automatic inversion of negatives



© University of Manchester

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



© University of Manchester

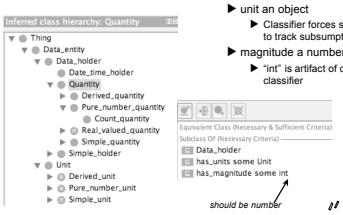
Quantities

- 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.

© University of Manchester

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



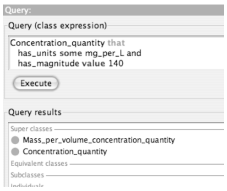
© University of Manchester

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.

© University of Manchester

DL Query for previous



© University of Manchester

Quantities and Units
which is "primitive"?

- Requirements
  - Avoid maintaining the hierarchies of quantities and nits 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

© University of Manchester

DL Queries

© University of Manchester

Example of use

- "Hemoglobin 13 mg%"
  - Serum_hemoglobin THAT
    has_value SOME (Concentration THAT
    has_magnitude VALUE 13
    has_units SOME mg_per_cent

© University of Manchester

Extensions to quantities

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

© University of Manchester

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”***