

# When and Why to use a Classifier?

**Alan Rector**

with acknowledgement to

**Jeremy Rogers, Pieter Zanstra, & the GALEN Consortium**

**Nick Drummond, Matthew Horridge, Hai Wang in CO-ODE/HyOntUSE**

Information Management Group Dept of Computer Science, U Manchester

Holger Knublauch, Ray Fergerson, ... and the Protégé-Owl Team

[rector@cs.man.ac.uk](mailto:rector@cs.man.ac.uk)

[co-ode-admin@cs.man.ac.uk](mailto:co-ode-admin@cs.man.ac.uk)

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

[protege.stanford.org](http://protege.stanford.org)

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

# Reasons to classify (1)

- **Managing Compositional ontologies / Terminologies**
  - “Conceptual Lego”
    - **Managing combinatorial explosions - the exploding bicycle**
  - **Empowering users**
    - “Just in time” ontologies
      - Give the users the Lego set with limited connectors
  - **Organising polyhierarchies / Modularizing ontologies**
    - “Normalising ontologies”
    - *Multiaxial indexing of resources*
  - **Providing multiple views -**
    - *Reorganising the ontology by new abstractions*
  - **Constraining ontologies & schemas**
    - **Enforcing constraints**
    - **Imposing policies**
      - **For clinical Statements with SNOMED entries**  
**in request mode    context is request**

## Reasons to classify(2)

- 'Matching' instances against classes
  - Resource/service discovery
  - Self-describing storage
    - 'Archetypes' & templates
- Providing a skeleton for default reasoning & Prototypes (but not to do the reasoning itself)
  - Molluscs typically have shells
    - Cephalopods are kinds of Molluscs but typically do not have shells
      - Nautiloids are kinds of Cephalopods but typically do have shells
        - » Nautilus ancestor are kinds of Nautiloids but do (did) not have shells
  - Biology is full of exceptions

# Classification is about Classes

- **Classification works for**
  - **Organising & constraining classes / schemas**
  - **Identifying the classes to which an instance definitely belongs**
    - **Or those to which it cannot belong**
- **Classification is open world**
  - **Negation as unsatisfiability**
    - **'not' == 'impossible' ("unsatisfiable")**
  - **Databases, logic programming, PAL, queries etc are closed world**
    - **Negation as failure**
      - **'not' == cannot be found**

# Reasons not to Classify

- **To query large number of instances**
  - **Open world (“A-Box”) reasoning does not work over large numbers of instances**
- **If the question is closed world**
  - **E.g. “Drugs licensed for treatment of asthma”**
- **If the query requires non-DL reasoning**
  - **E.g. numerical, optimisation, probabilistic, ...**
    - **Would like to have a more powerful hybrid reasoner**
- **For Metadata and Higher Order Information**
  - **Classifiers are strictly first order**
    - **A few things can be ‘kluged’**
- **If there are complex defaults and exceptions**
  - **“Prototypical Knowledge”**
    - **E.g. “Molluscs typically have shells”**
      - **NB Simple exceptions can be handled, but requires care**

# Use instead

- **To query large numbers of instances OR  
If the query is closed world**
  - **Queries / constraints over databases**
  - **Instance stores / triple stores / ...**
  - **Rules**
    - **DL-programming**
    - **JESS, Algernon, Prolog, ...**
  - **Belief revision / non-monotonic reasoning**
- **If query requires Non DL Reasoning**
  - **Hybrid reasoners or ?SWRL?**
    - **No good examples at the moment**
- **For defaults and Exceptions & Prototypical Knowledge**
  - **Traditional frame systems**
    - **More expressive default structure than Protégé**
      - **Exceptions for classes as well as instances**
        - » **Over-riding rather than narrowing**

# Classification to build Ontologies: Conceptual Lego



hand

extremity

body

chronic

acute

abnormal

normal

ischaemic

deletion

polymorphism

gene

protein

cell

expression

Lung

inflammation

infection

bacterial

# Logic-based Ontologies: Conceptual Lego

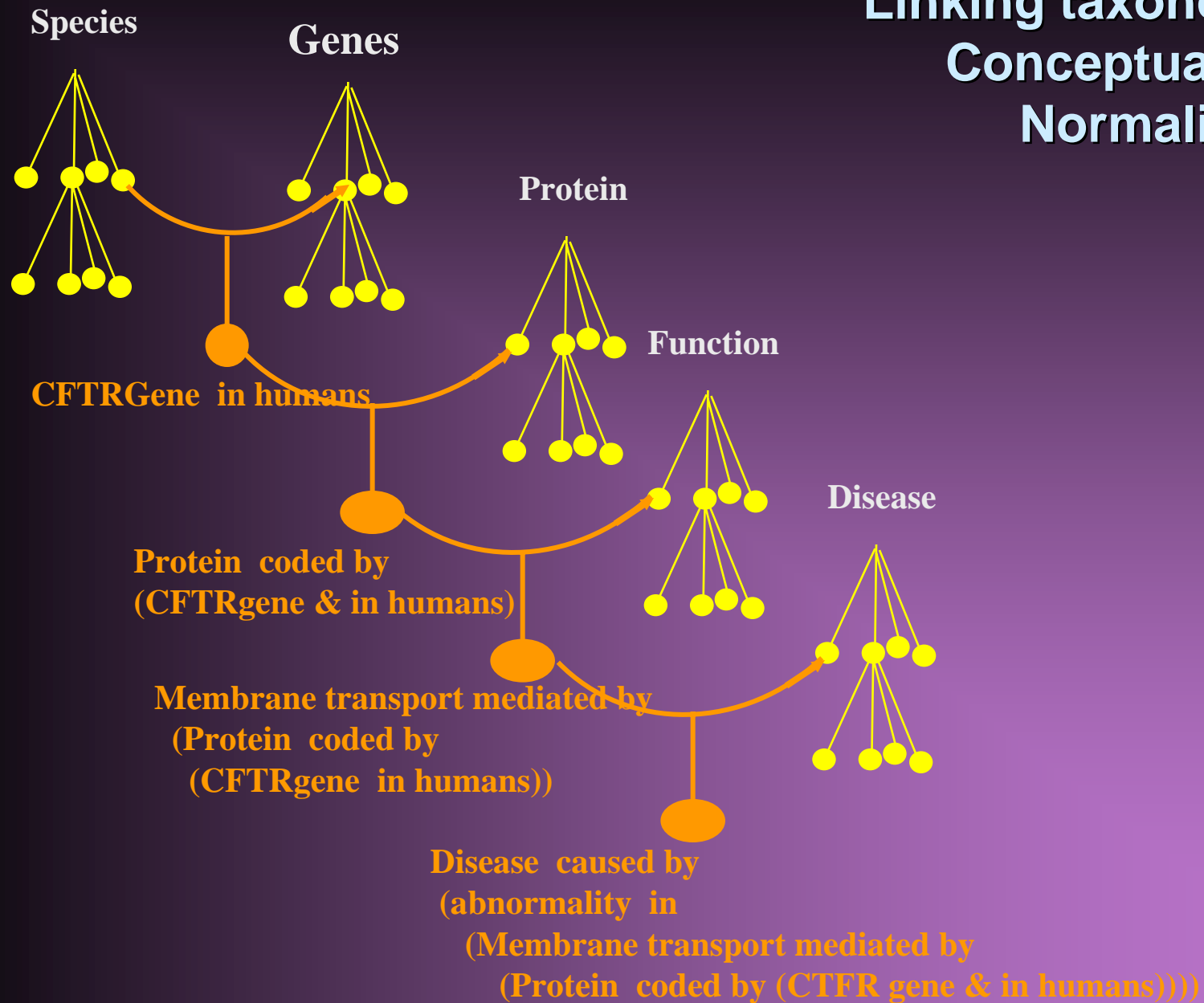
“SNPolymorphism of CFTRGene causing Defect in MembraneTransport of ChlorideIon causing Increase in Viscosity of Mucus in CysticFibrosis...”



“Hand which is  
anatomically  
normal”

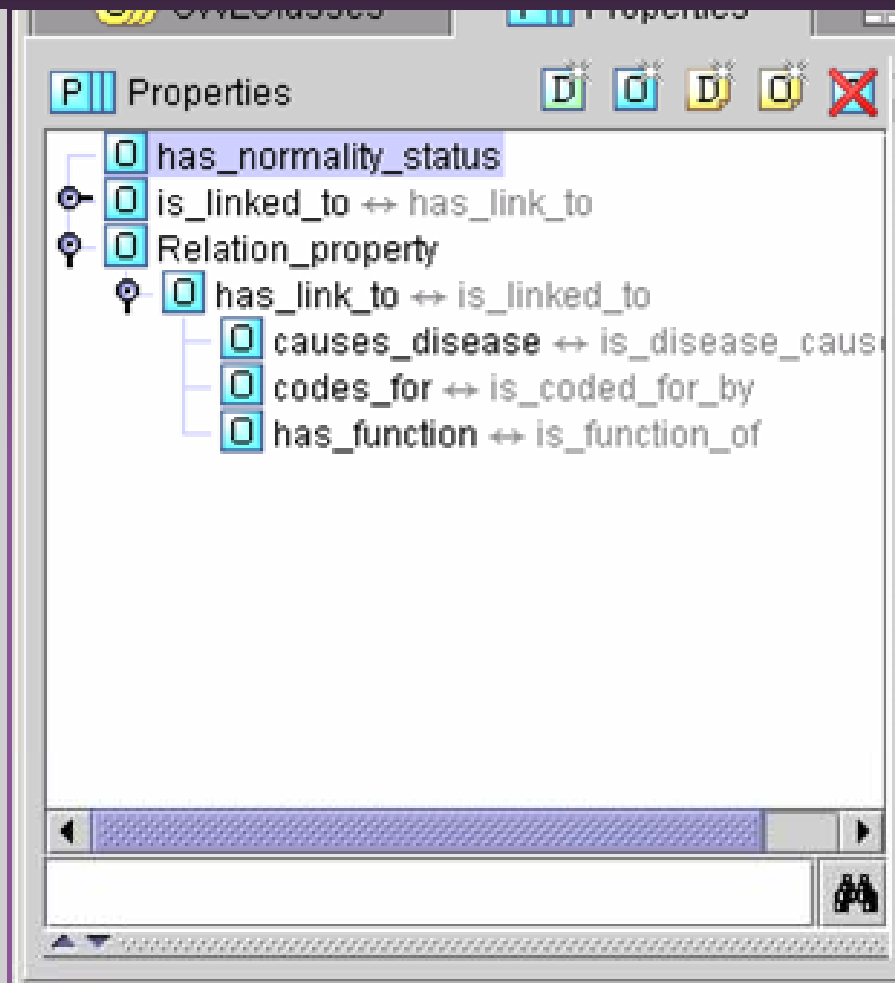
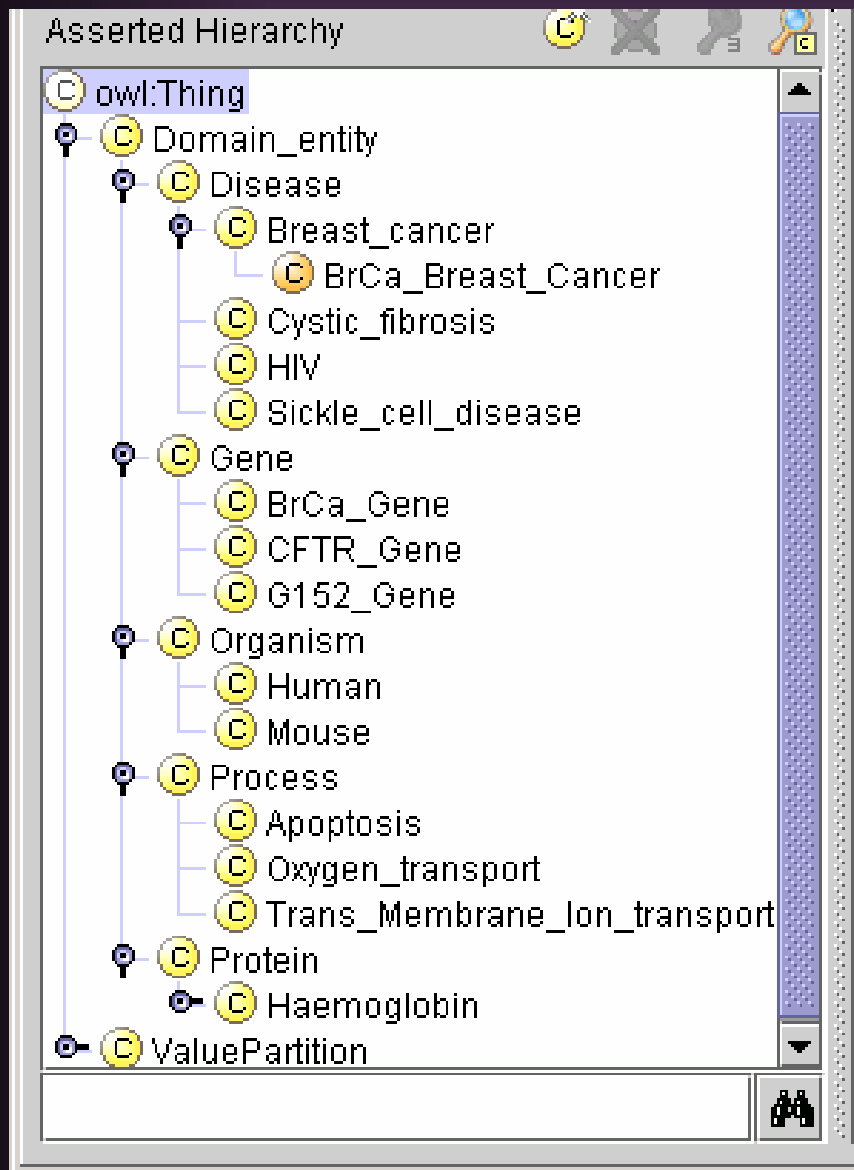


# Linking taxonomies: Conceptual Lego Normalisation



# Conceptual Lego and Normalisation Practical Example

# Take a Few Simple Concepts & Properties



# Combine them in Descriptions which can be simple....

The screenshot shows a web ontology editor interface. On the left is a class hierarchy tree under 'owl:Thing'. The 'Disease' class is expanded, showing subclasses: 'Breast\_cancer', 'Cystic\_fibrosis', 'HIV', and 'Sickle\_cell\_disease'. 'Sickle\_cell\_disease' is selected. On the right, the 'Sickle\_cell\_disease' class is being edited. The 'rdfs:comment' field is empty. Below, the 'Asserted Conditions' tab is active, showing a list of conditions. The first condition is 'Disease' with a 'NECESSARY & SUFFICIENT' relationship. The second condition is '∃ is\_disease\_caused\_by Sickling\_haemoglobin' with a 'NECESSARY' relationship. Both conditions have a yellow 'E' button next to them.

**Sickle cell disease is a disease caused  
*some* sickling haemoglobin**

or which can be as complex as you like

The screenshot shows an OWL editor interface. On the left, a class hierarchy is displayed under 'owl:Thing', including 'Domain\_entity', 'Disease', 'Breast\_cancer', 'BrCa\_Breast\_Cancer', 'Cystic\_fibrosis', 'HIV', 'Sickle\_cell\_disease', 'Gene', 'BrCa\_Gene', 'CFTR\_Gene', 'G152\_Gene', 'Organism', 'Human', 'Mouse', and 'Process'. The 'Cystic\_fibrosis' class is highlighted. In the center, a text field contains 'Cystic\_fibrosis' and 'rdfs:comment'. On the right, an 'Edit OWL Expression' dialog box is open, containing the following OWL expression:

```
∃ is_disease_caused_by (Trans_Membrane_Ion_transport ∩  
  ∃ has_normality_status nonNormal ∩  
  ∃ is_function_of (Protein ∩  
    ∃ is_coded_for_by CFTR_Gene))
```

**Cystic fibrosis is caused by some non-normal ion transport that is the function of a protein coded for by a CFTR gene**

# Add some definitions

owl:Thing

- Domain\_entity
  - Disease
    - Breast\_cancer
    - Cystic\_fibrosis
    - Diseases\_linked\_to\_CFTR\_Genes**
    - Diseases\_linked\_to\_Genes
    - HIV
    - Haemoglobinopathy
    - Sickle\_cell\_disease
  - Gene
    - BrCa\_Gene
    - CFTR\_Gene
    - G152\_Gene
  - Organism
    - Human

Diseases\_linked\_to\_CFTR\_genes

rdfs:comment

Asserted Inferred Abstract Syntax Abstra:

Asserted Conditions

NECESSARY & SUFFICIENT

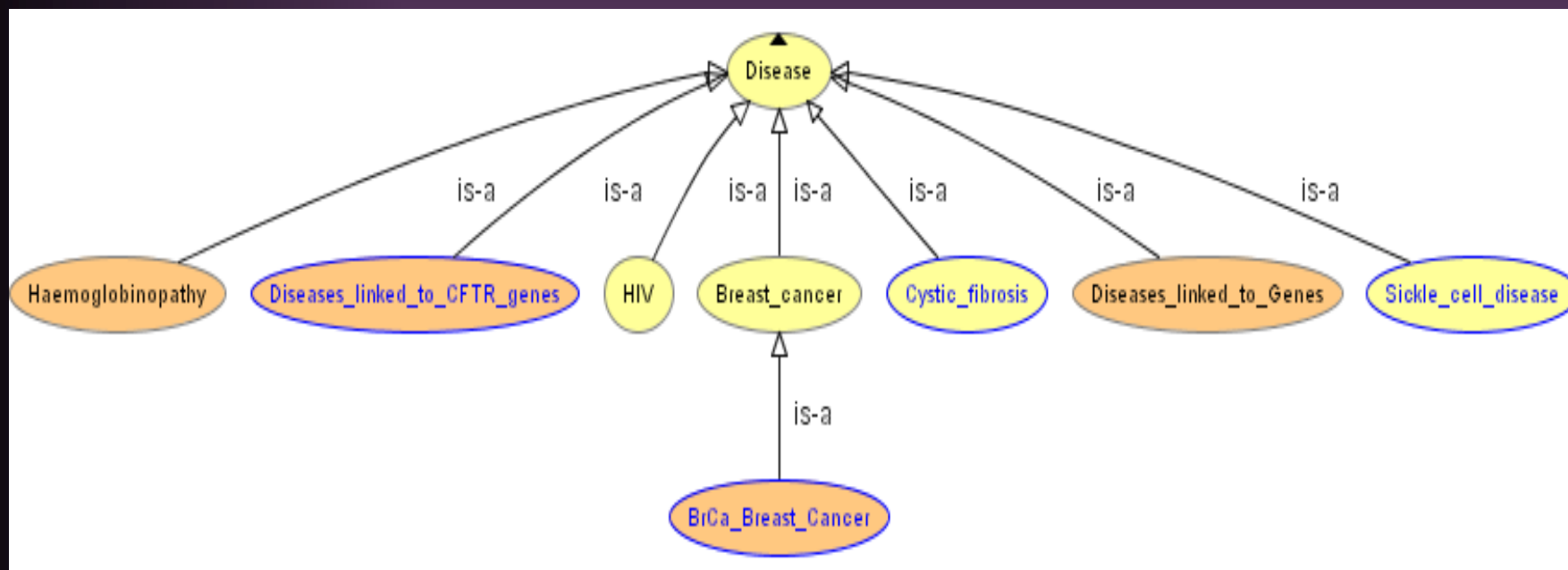
Disease

$\exists$  is\_linked\_to CFTR\_Gene

NECESSARY

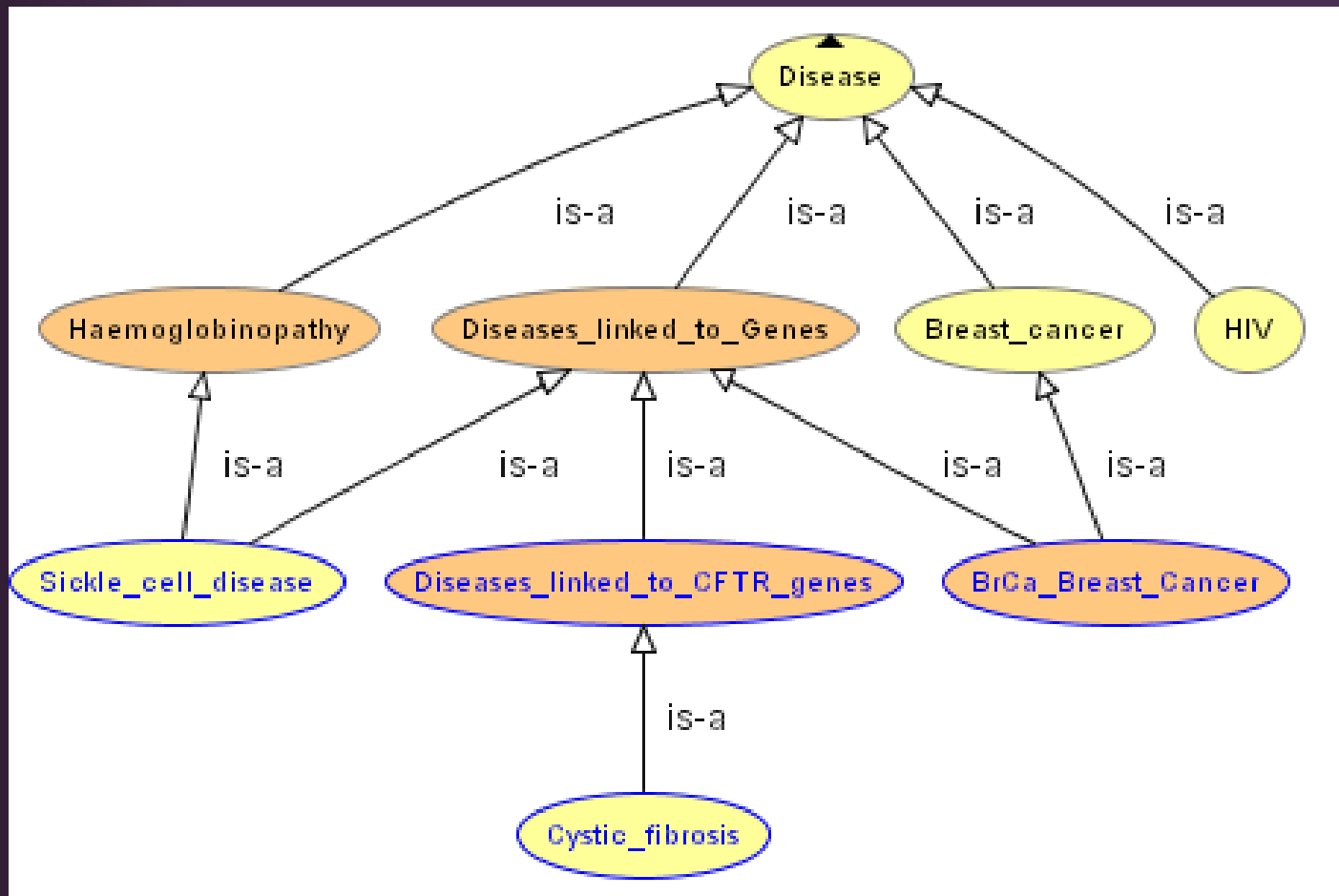
**“Diseases linked to CFTR Genes”**

# We have built a simple tree



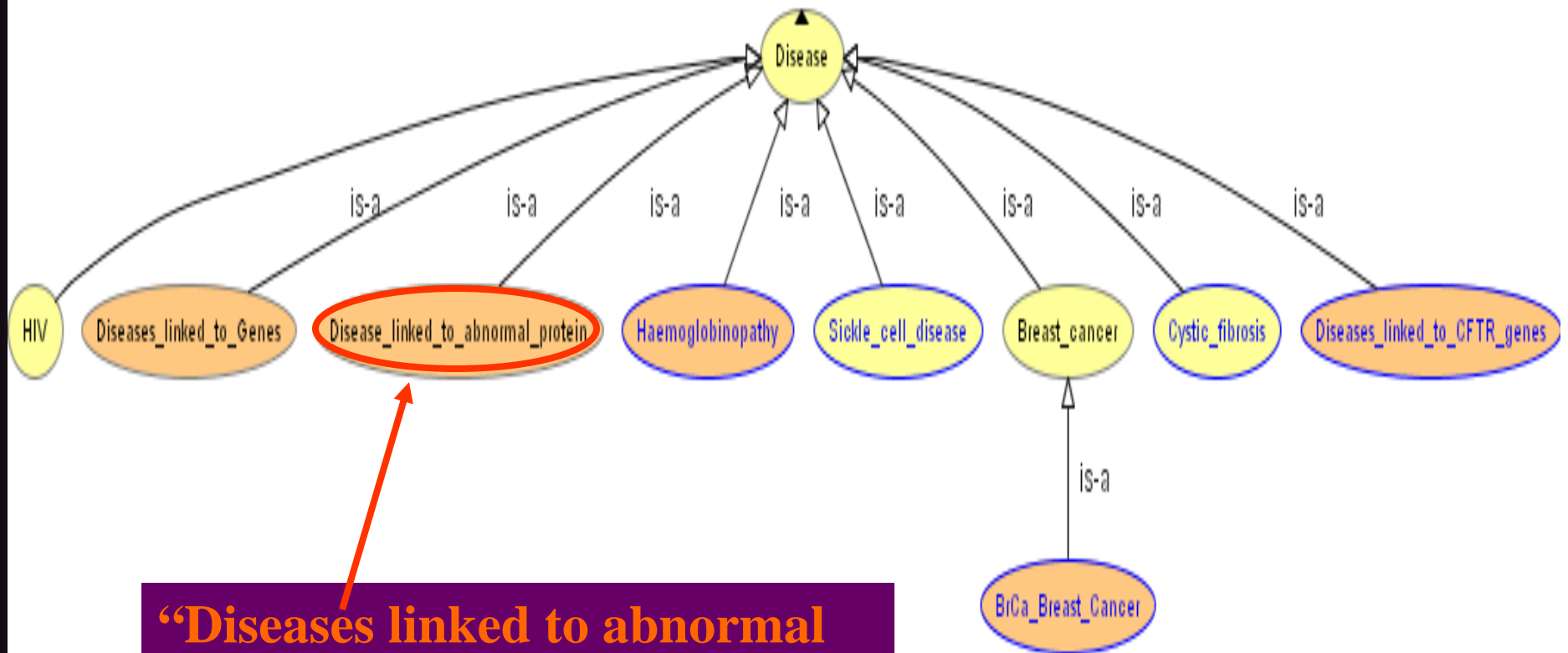
easy to maintain

# Let the classifier organise it



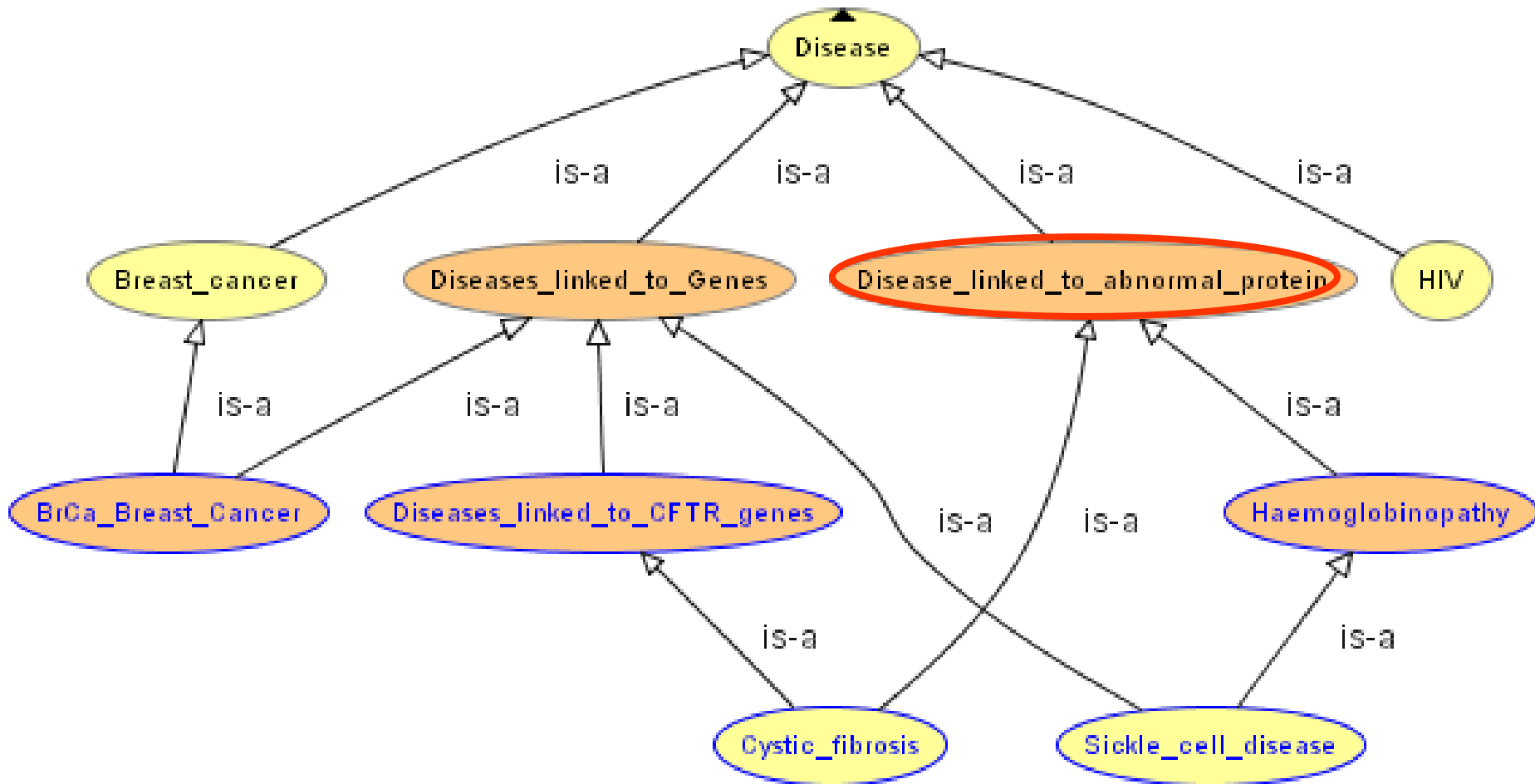


If you want more abstractions,  
just add new definitions  
(re-use existing data)

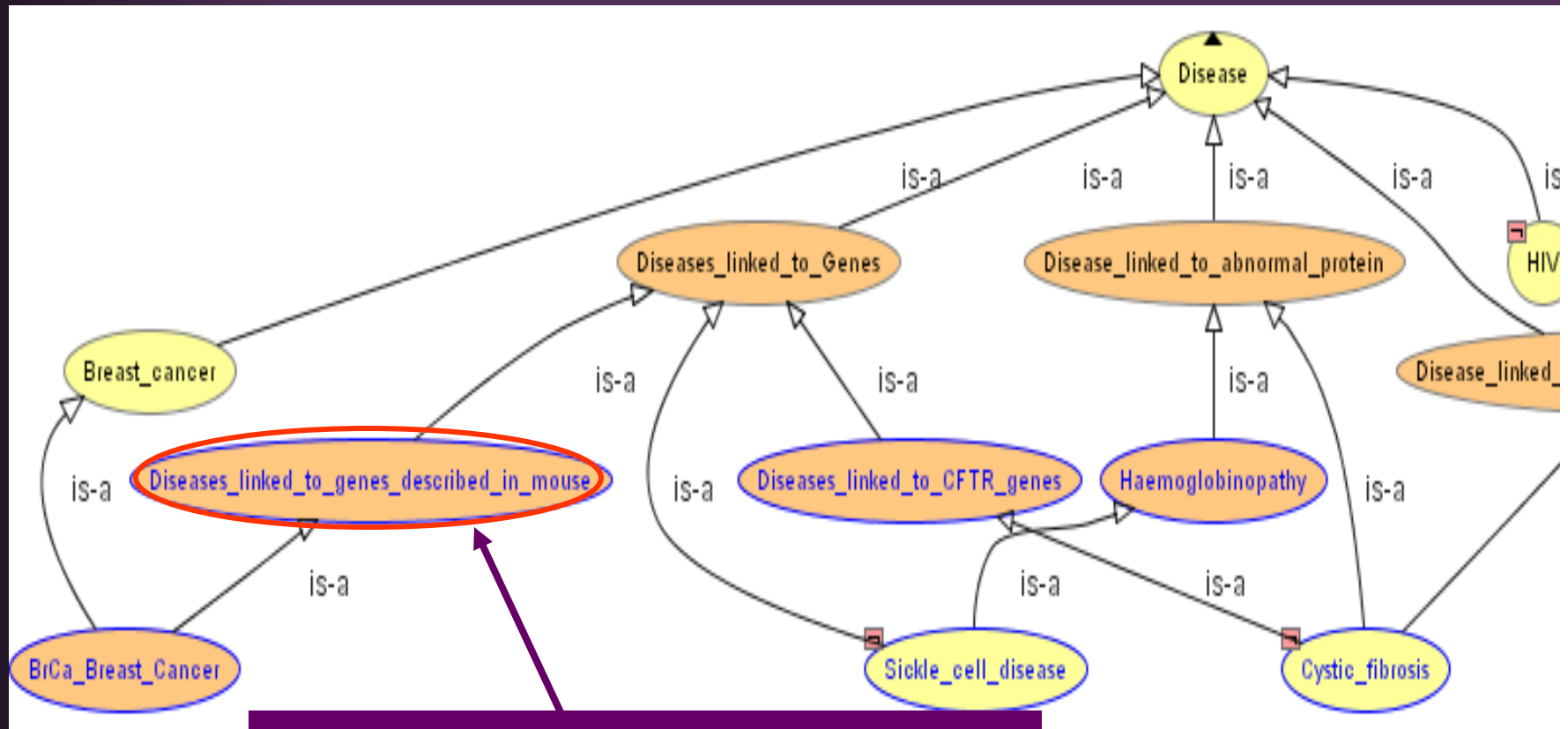


**“Diseases linked to abnormal proteins”**

# And let the classifier work again



# And again – For a view based on species



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

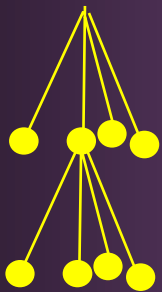
# And let classifier check consistency (My first try wasn't)

The screenshot shows the Protege OWL editor interface. On the left is a class hierarchy tree with 'Cystic\_fibrosis' selected. The main workspace displays the 'Cystic\_fibrosis' class with its 'rdfs:comment' field. Below this, the 'Asserted Conditions' tab is active, showing a logical expression:  $(\exists \text{is\_disease\_caused\_by Trans\_Membrane\_Ion\_transport}) \sqcap (\exists \text{has\_normal})$ . A tooltip explains that this represents the intersection of three conditions: any object where some instances are disease caused by Trans Membrane Ion transport, any object which has a non normal as its normal, and any object where some instances are function of protein. At the bottom, a table lists 'Changed superclasses' for various classes, with 'Cystic\_fibrosis' marked as 'Inconsistent'.

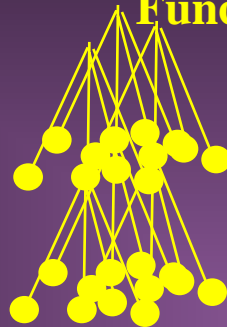
Class	Changed superclasses
BrCa_Breast_Cancer	Added Diseases_linked_to_genes_described_in_mouse
Cystic_fibrosis	Inconsistent
Diseases_linked_to_CFTR_gene	Moved from Disease to Diseases_linked_to_Genes
Diseases_linked_to_genes_described_in_mouse	Moved from Disease to Diseases_linked_to_Genes
Haemoglobinopathy	Moved from Disease to Disease_linked_to_abnormal_protein

# Normalising (untangling) Ontologies

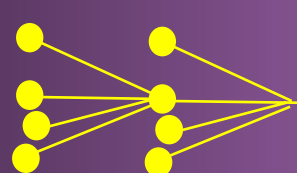
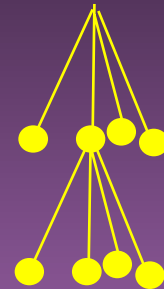
Structure



Structure  
Part-whole  
Function



Function



Part-whole

# Untangling and Enrichment

Using a classifier to make life easier

Substance

- Protein

- - ProteinHormone

- - - Insulin

- - Enzyme

- - - ATPase

- Steroid

- - SteroidHormone^

- - - Cortisol

-Hormone

- - ProteinHormone^

- - - Insulin^

- - SteroidHormone^

- - - Cortisol^

- Catalyst

- - Enzyme^

- - - ATPase^

# Normalisation & Quality Assurance

- Humans recognise errors of commission easily
  - Miss errors of omission
- Classifiers convert errors of omission to errors of commission
  - Inadequate definitions create “orphans”
    - BodyPart
      - Parts of Heart
        - Ventricle
        - ...
        - CardiacSeptum
- Classifiers flag errors of commission
  - Over definition leads to inconsistency (unsatisfiability)
    - “Pneumonia located in the brain”

# Enforcing constraints & policies

- A class with both necessary & sufficient and additional necessary conditions acts as a rule
- The Unit testing Framework supports checking that rules are enforced

The screenshot displays a software interface for defining and managing rules. At the top, a text input field contains the identifier "Request\_CD\_ACT". Below it, a search field labeled "rdfs:comment" contains the text "Any Act with a mood code request must have a Context Code request." To the right, a partial view of a tree structure shows "rdfs:co".

Below the search field, there are two tabs: "Asserted" (which is selected) and "Inferred". Under the "Asserted" tab, there is a section titled "Asserted Conditions". This section contains a list of conditions with associated icons and labels:

- A yellow circle icon with a plus sign, labeled "NECESSARY & SUFFICIENT".
- A yellow circle icon with a plus sign, labeled "hl7:Act".
- A yellow circle icon with a plus sign, labeled "∃ has\_CD CD\_code\_holder\_for\_request".
- A yellow circle icon with a plus sign, labeled "NECESSARY".
- A yellow circle icon with a plus sign, labeled "∃ has\_mood hl7:Request\_mode\_code".

On the right side of the "Asserted Conditions" list, there are two orange buttons: one with a hamburger menu icon (≡) and one with a yellow background and a black border containing the letter "E".

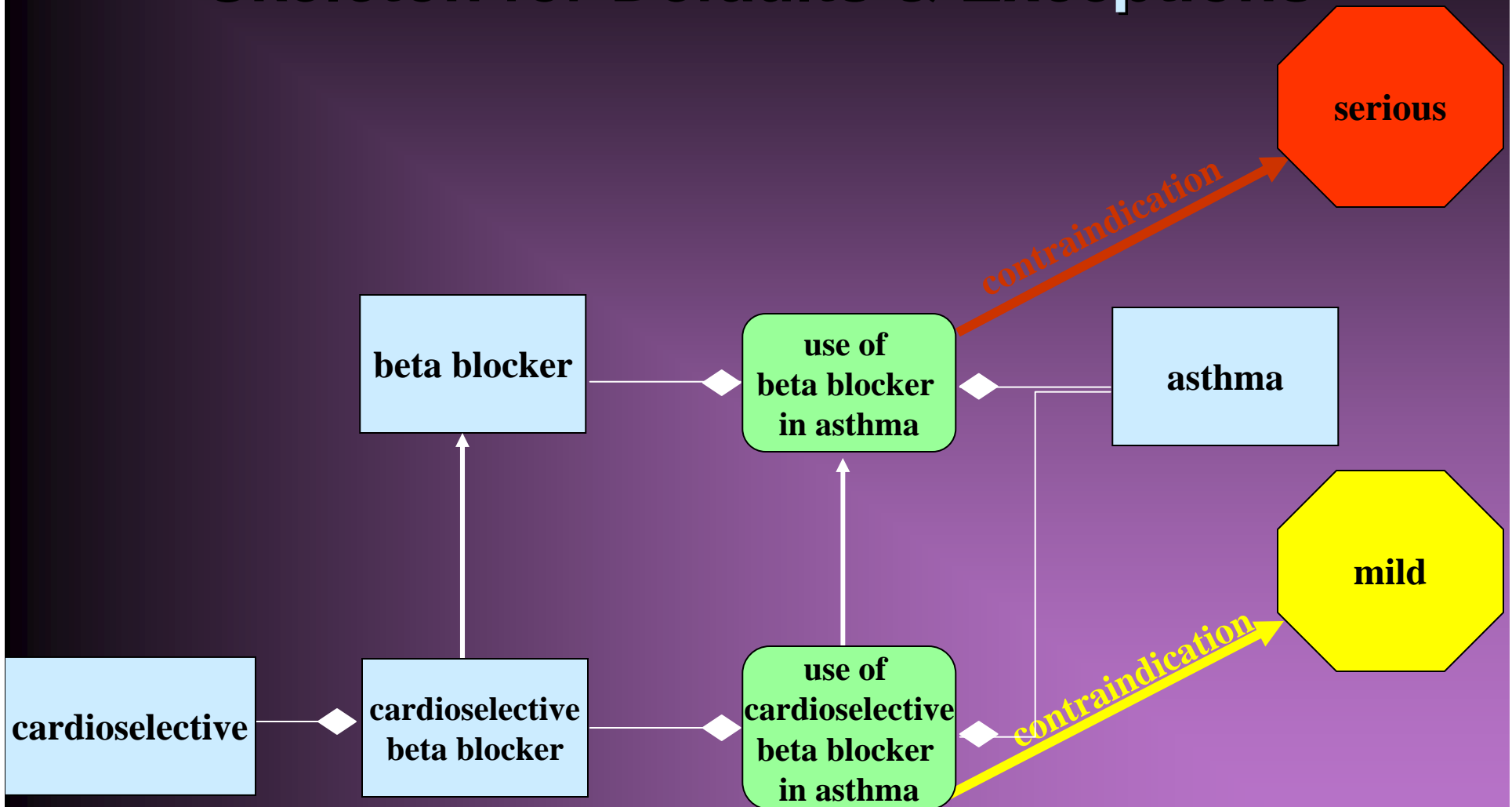


# A Probe class to check a constraint

## All Tests Passed

Type	Source	Test Result
✓	Probe_Request_mood_request_context_done	Expected consistency (inconsistent) achieved
✓	Probe_Act_context_request_mode_event	Expected consistency (inconsistent) achieved

# Skeleton for Defaults & Exceptions



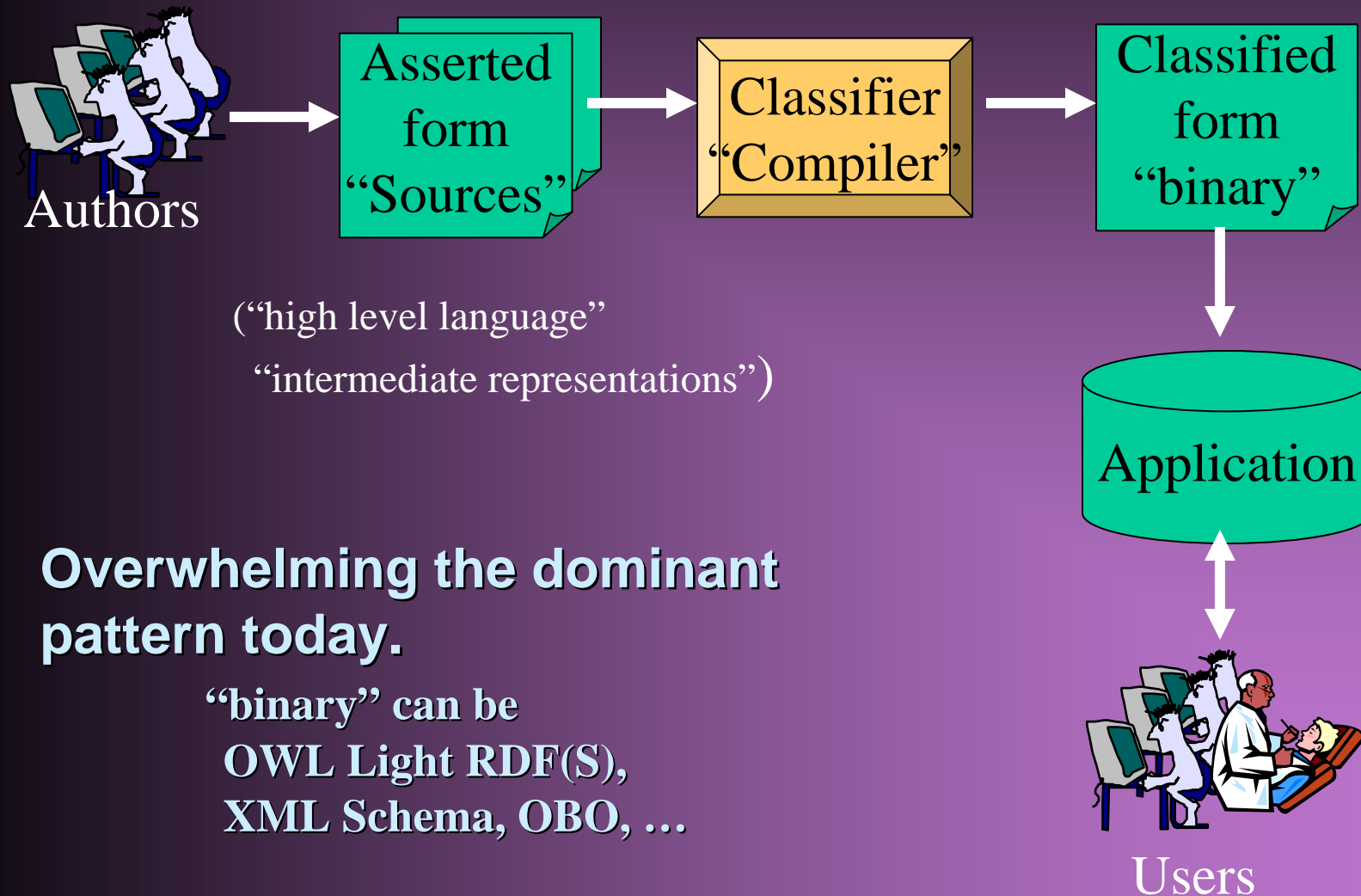
Experience: Normalised ontologists lead to clean default inheritance

# When to Classify

- ... but isn't having a classifier an intollerable overhead for the applications?
  - It depends on the life cycle you choose
- Life cycles
  - Pre-coordination
  - Just in time coordination
  - Post Coordination

# Pre-coordination

If the terms can be enumerated in advance



Overwhelming the dominant pattern today.

“binary” can be  
OWL Light RDF(S),  
XML Schema, OBO, ...

# Commit Results to a Pre-Coordinated Ontology

The screenshot displays an ontology editor interface. At the top, there are two panels: 'Asserted Hierarchy' and 'Inferred Hierarchy', both showing a tree structure starting with 'owl:Thing' and including subclasses like 'Domain\_entity', 'Probe\_Catgories\_for\_demo\_and\_test', 'ValuePartition', and 'ValueSelector'. Below these is a 'Classification Results' table with two columns: 'Class' and 'Changed superclasses'. The table lists several classes and their updated superclasses. An orange arrow points from the text 'Assert ("Commit") changes inferred by classifier' to the 'Assert selected change(s)' button in the interface.

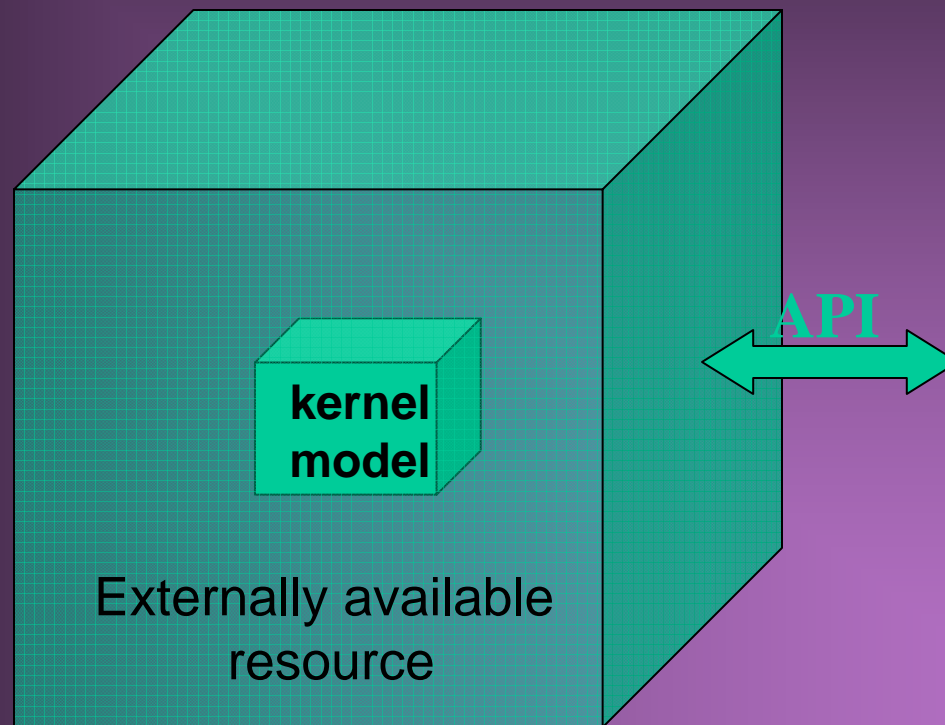
Class	Changed superclasses
Aortic_stenosis	Added Disorder_of_FMA_heart
Aortic_valve	Added Structural_parts_of_heart
Cardiac_chamber	Added Structural_parts_of_heart
Disorder_of_FMA_heart	Moved from Disorder, Probe_Catgories_for_demo_a...
Half_heart	Added Structural_parts_of_heart
Pericarditis	Added Disorder_of_Clinical_heart
Pericardium	Added Clinical_parts_of_heart

Assert  
("Commit")  
changes  
inferred by  
classifier

# Post coordination

When there are combinatorially many potential terms

- “Lazy classification” on demand



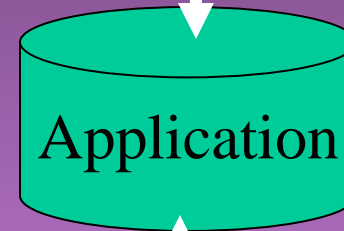
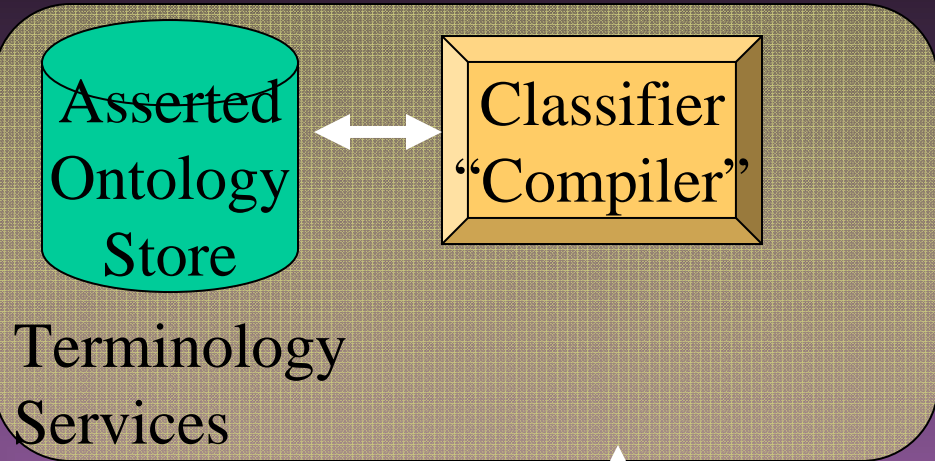
*Big on the outside: small kernel on the inside*  
*Avoid the exploding bicycle*

# Post Coordination



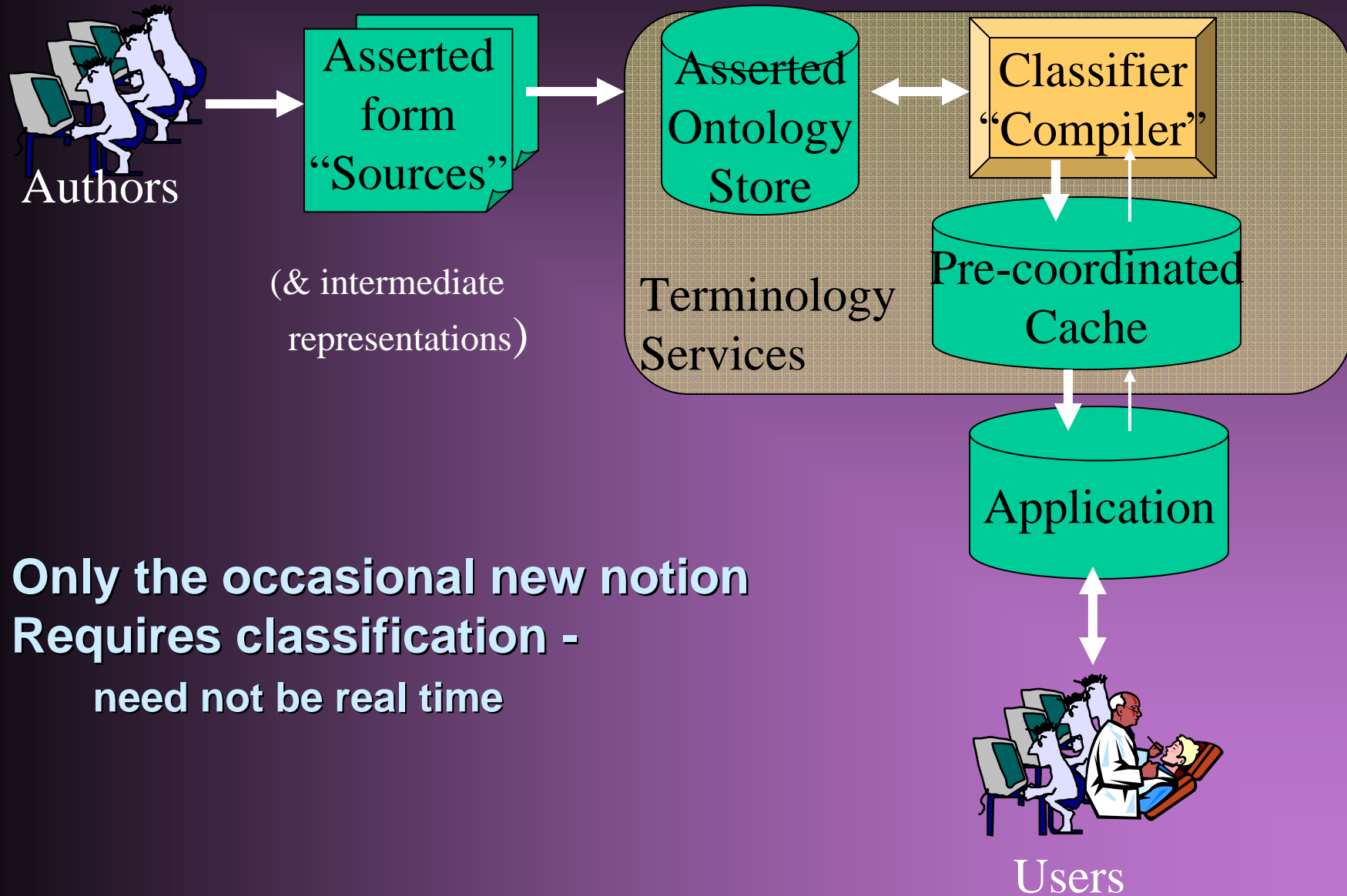
Asserted  
form  
“Sources”

(& intermediate  
representations)



**But requires a classifier  
available at application time**

# A Compromise Just in Time Coordination





# Summary

- **Why Classify**
  - **Managing Compositional ontologies / Terminologies**
    - “Conceptual Lego”
    - Empowering users - just in time classification
    - Providing views & deferring decisions on abstractions
    - Quality assurance
  - **Constraining concepts & schemas**
  - **Providing a skeleton for default reasoning & Prototypes**
- **When to classify**
  - **Pre-coordination**
  - **Just-in-time coordination**
  - **Post-coordination**

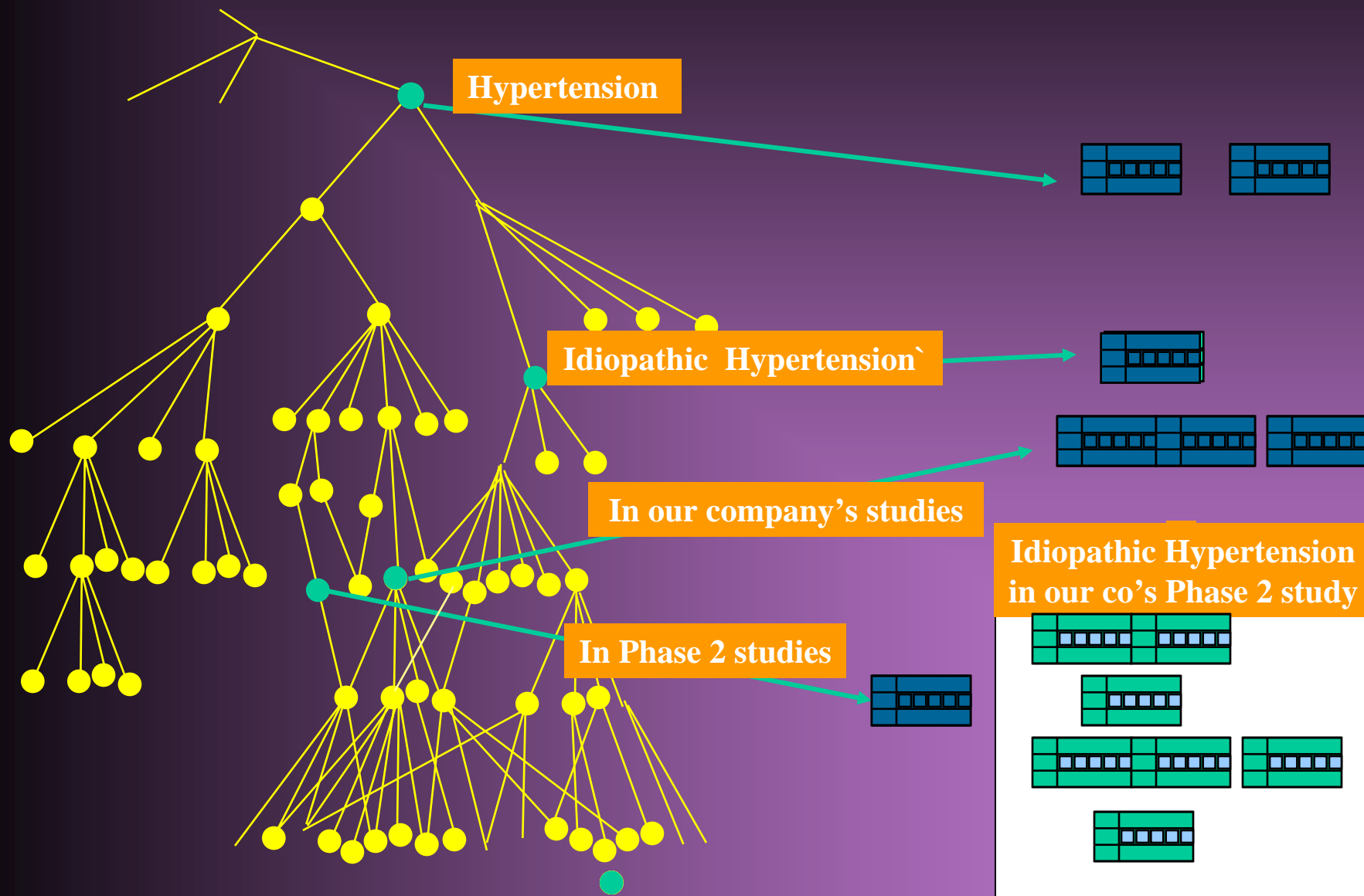
# Summary: When to Classify?

*Applications do not need a classifier to benefit from classification*

- **Pre-coordination**
  - **If concepts/terms can be predicted**
  - **When classifier is not available at run time**
  - **When we must fit with legacy applications**
- **Post-coordination**
  - **When a a few concepts are needed from a large potential set**
  - **When a classifier is available**
    - **and time cost is acceptable**
  - **When applications can be built or adapted to take advantage**



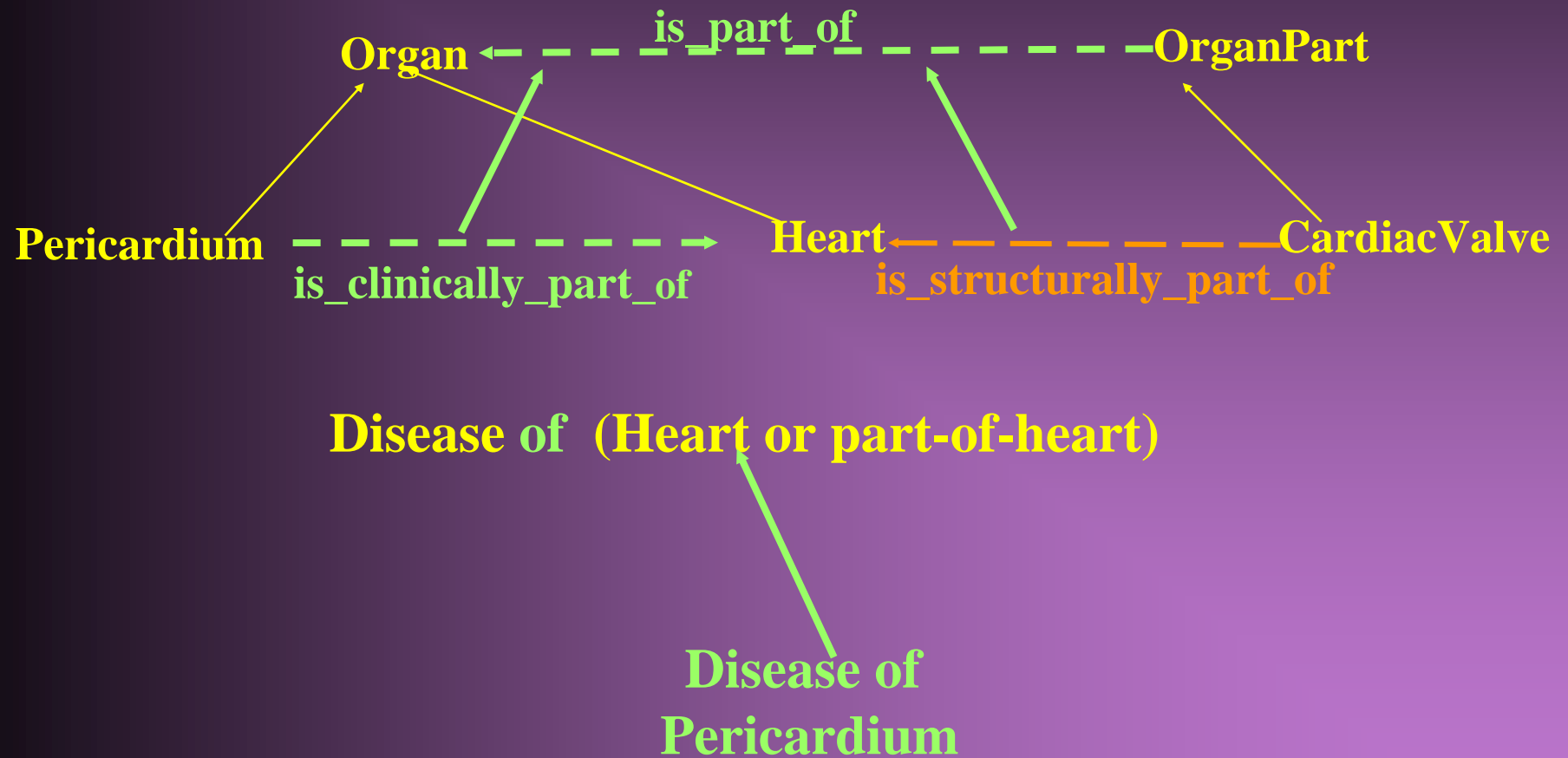
# Skeleton for Fractal tailoring forms for clinical trials



# More on Views

- **A problem in the digital anatomist**
  - **To an anatomist, the Pericardium and the Heart are separate organs**
  - **To a clinician they are part of the same organ**
    - **A disease of the pericardium counts as a kind of heart disease**

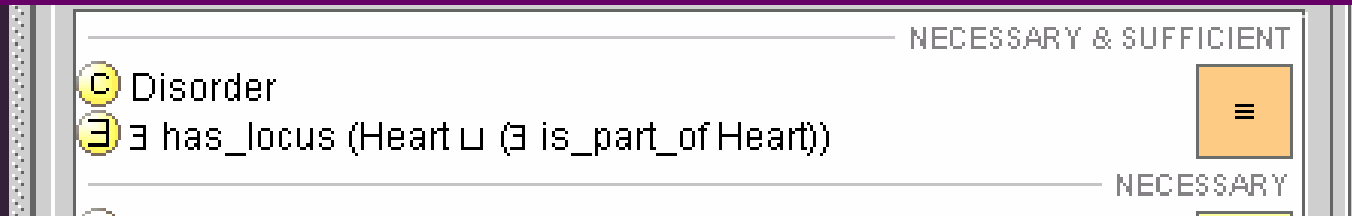
# Represent context and views by variant properties



# Protégé-OWL alternative views

## Disorder of “Clinical heart”

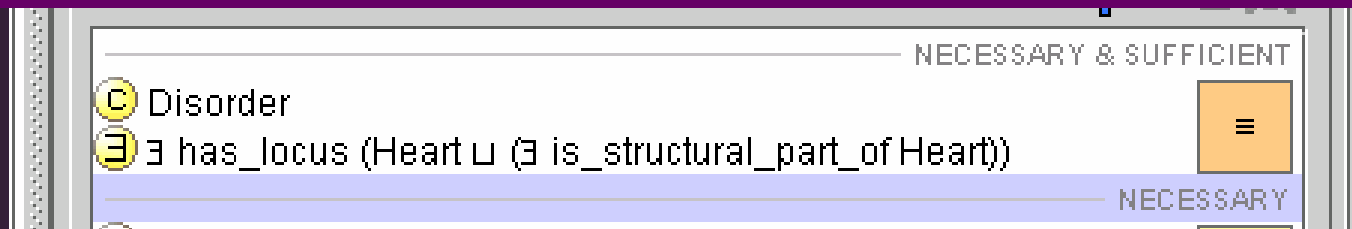
“Disorder of heart of any part of the heart”  
(including *clinical* and *functional* parts)



A screenshot of the Protégé-OWL interface showing a class definition for 'Disorder'. The class is defined as 'NECESSARY & SUFFICIENT' and is associated with the property 'has\_locus'. The domain is 'Heart' and the range is '∃ is\_part\_of Heart'. The interface includes a yellow 'C' icon for the class name and a yellow 'E' icon for the property. A yellow button with a menu icon is visible on the right side of the definition area.

## Disorder of “FMA heart”

“Disorder of heart or any *structural* part of the heart”



A screenshot of the Protégé-OWL interface showing a class definition for 'Disorder'. The class is defined as 'NECESSARY & SUFFICIENT' and is associated with the property 'has\_locus'. The domain is 'Heart' and the range is '∃ is\_structural\_part\_of Heart'. The interface includes a yellow 'C' icon for the class name and a yellow 'E' icon for the property. A yellow button with a menu icon is visible on the right side of the definition area.